LAUREA

Elinkaarimallien soveltaminen tietokanta-pohjaisen web-sovelluksen kehittämisessä

CASE: Taitouinnin pistelaskujärjestelmä

Kuhmonen Timo, Nikkanen Juuso

2011 Leppävaara

Laurea-ammattikorkeakoulu
Laurea Leppävaara

Elinkaarimallien soveltaminen tietokantapohjaisen web-sovelluksen kehittämisessä

CASE: Taitouinnin pistelaskujärjestelmä

Kuhmonen, Timo & Nikkanen, Juuso
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Helmikuu, 2011

Laurea-ammattikorkeakoulu **Tiivistelmä**
Laurea Leppävaara
Tietojenkäsittelyn koulutusohjelma

Kuhmonen, Timo & Nikkanen, Juuso

Elinkaarimallien soveltaminen tietokantapohjaisen web-sovelluksen kehittämisessä

Vuosi 2011 Sivumäärä 68

Tämän opinnäytetyön tavoitteena on esitellä ja tutkia tietojärjestelmän elinkaarimalleja sekä toteuttaa teorian perusteella Suomen Uimaliitolle taitouinnin pistelaskujärjestelmä. Työssä tarkastellaan myös ohjelmistokehityksen vaiheita kirjallisuuden ja kehitetyn järjestelmän pohjalta.

Lähtötilanteessa Suomen Uimaliitolla ei ollut käytössään nykyaikaista automatisoitua taitouintikilpailun pisteytysjärjestelmää. Lopputulosten tuottaminen manuaalisesti oli aikaisemmin taulukkolaskennan varassa, minkä takia inhimillisten virheiden mahdollisuus oli suuri. Lisäksi kilpailussa käytettävät tulosteet eivät olleet yhdenmukaisia ja niiden saanti oli toisinaan hidasta.

Esitutkimuksen aikana käydyssä haastattelussa ilmeni, että järjestelmän kehittäminen toisi kilpailutoimintaan suoraviivaisuutta ja selkeyttä. Projektin toteutukseen ei asetettu tarkkoja vaatimuksia tai rajoituksia, minkä takia järjestelmä voitiin kehittää vapaasti opinnäytetyön tutkimusten pohjalta. Pistelaskujärjestelmä rakennettiin pääosin PHP-ohjelmointikielellä ja MySQL-tietokantaohjelmistolla.

Opinnäytetyössä käytettiin konstruktiivista tutkimusotetta, jossa asiakkaan ongelma ratkaistiin haastatteluiden ja ohjelmistokehityksen teorian avulla. Työ koostui sekä teoreettisesta tutkimusosiosta että konstruktiona syntyneen pistelaskujärjestelmän suunnittelemisesta ja tuottamisesta. Järjestelmän kehittämisessä käytettäväksi elinkaarimalliksi valikoitui prototyyppimalli.

Asiasanat Taitouinti, PHP, MySQL, elinkaarimalli, tietojärjestelmä

Laurea University of Applied Sciences **Abstract**
Laurea Leppävaara
Bachelor's Degree Programme in Business Information Systems

Kuhmonen, Timo & Nikkanen, Juuso

Adaptive use of life cycle models while developing Database-driven Web Application

Year 2011 Pages 68

The objective of this thesis is to present and research different life cycle models of information systems and to create a synchronized swimming point calculating system for the Finnish Swimming Association. The thesis also includes a description of the stages of system development based on literature and the information system which was created.

The Finnish Swimming Association did not have a modern information system for synchronized swimming before this thesis. The probability of human errors was considerable, because almost all methods and actions were made manually on excel. The results and other printable data had variable appearances and the availability of this data was limited.

The project was started with a feasibility study. Studies showed that a new synchronized swimming point calculating system would offer a clearer and more linear way to survive the challenges of the competition than earlier methods could. Strict requirements or restrictions were not assigned so the application developed could be created based on the thesis research. The system was created mainly using the PHP programming language and MySQL database.

The method used in this thesis is the constructive research method. It aims at solving a practical problem by interviews and by using theory based information. The output of the thesis consists of a theoretical research section and the designing and producing of the construction: a synchronized swimming point calculating system. The lifecycle model used in this project was a prototype model.

Key words Synchronized swimming, PHP, MySQL, Life cycle model, Information system

Sisällys

1 Johdanto 7
2 Tavoite ja menetelmät 8
3 Taustatietoa 8
4 Käsitteistö 9
4.1 Käyttöliittymätason käsitteistö 10
4.2 Sovellustason käsitteistöä 10
4.3 Tietokantatason käsitteistöä 12
5 Elinkaarimallit 13
5.1 Vesiputousmalli 14
5.2 Prototyyppilähestymistapa 15
5.3 Spiraalimalli 16
5.4 Evo-malli 17
5.5 RUP 17
6 Tietojärjestelmän kehittämisen vaiheet 19
6.1 Esitutkimus 20
6.2 Vaatimusmäärittely 21
6.3 Kuvausmenetelmät 22
6.3.1 Toimintojen ja tietojen kuvaaminen 22
6.3.2 Luokkakaaviot 23
6.4 Järjestelmäanalyysi 23
6.5 Suunnittelu 24
6.6 Toteutus 24
6.7 Testaus 25
6.8 Käyttöönotto 26
6.9 Jatkokehitys ja ylläpito 26
7 Case: Taitouinnin pistelaskujärjestelmä 27
7.1 Projektin vaiheet 27
7.1.1 Esitutkimus 27
7.1.2 Määrittely ja suunnittelu 28
7.1.3 Toteutus 29
7.1.4 Testaus ja käyttöönotto 30
7.2 Kysymyslomake 32
7.3 Kyselyn vastausten analysointi 33
7.4 PHP & MySQL valinta 34
7.5 Tietokannan rakenne 34
7.6 Järjestelmän ulkoasu 38
7.7 Aikataulu 39

7.8 Riskit 39
8 Yhteenveto ja johtopäätökset 40
Liite 1: Vaatimusmäärittelyraportti 47
Liite 2: Kysymyslomake 57
Liite 3: Käyttötapauskuvaukset 58
Liite 4: Kaaviot: 64
Liite 5: Esimerkki ohjelmistokoodista 67

1 Johdanto

Nykypäivän yhteiskunnassa vallitsee automatisoinnin trendi, joka pyrkii vähentämään manuaalisen työn aiheuttamaa taakkaa. Uudet tietojärjestelmät, jotka on räätälöity ja kehitetty vastaamaan juuri kunkin tilanteen tarpeita, ovatkin monesti varteenotettava tapa tehostaa ja nopeuttaa eri prosesseja samalla vähentäen inhimillisten riskien määrää. Ohjelmiston suunnittelu ja kehitys tarjoaakin lähes rajattomat mahdollisuudet uusien innovatiivisten ratkaisujen luomiseen esimerkiksi erilaisten elinkaarimallien käytön myötä. On siis erittäin tärkeää, että osataan valita tai räätälöidä oikeanlainen elinkaarimalli ohjelmiston kehitykselle, jotta olisi mahdollista päästä mahdollisimman hyvään lopputulokseen minimoiden projekteihin kuuluvat riskit.

Tässä opinnäytetyössä esitellään erilaisia ohjelmistokehityksen elinkaari- eli vaihejakomalleja sekä niiden soveltuvuutta pienen ohjelmiston kehitykseen. Elinkaarimallien sisällön ymmärtämiseksi on myös tärkeää ymmärtää käsitteen ”ohjelmistokehitys” sisältämät pienemmät osa-alueet. Tästä syystä olemme tutkineet ohjelmistokehityksen eri vaiheita siltä osin kun ne koskettavat luomamme järjestelmän kehitysvaiheita. Työssä käymme läpi myös lyhyesti ohjelmistokehityksen avainsanastoa, jotta aiheeseen perehtymätön lukija voi ymmärtää mistä oikeastaan on kyse.

Opinnäytetyön lähteinä on käytetty kirjallisuutta ja sähköisiä lähteitä. Nopeasti muuttuvan alan takia kirjallisuuden faktat vanhenevat nopeasti – paikoin jo ennen kirjan julkaisemista. Tästä johtuen olemme käyttäneet tutkimuksen apuna sosiaalisen median tukea kuten Internetin keskustelupalstoja.

2 Tavoite ja menetelmät

Tämän opinnäytetyön tavoitteena on luoda uusi taitouinnin pistelaskujärjestelmä Suomen Uimaliitolle. Lisäksi tutkittiin, kuinka pistelaskujärjestelmä tulisi rakentaa, jotta se vastaisi mahdollisimman tarkasti loppukäyttäjän tarpeita.

Opinnäytetyössä tarkastelimme, millaisia eri vaiheita tietojärjestelmien kehittämiseen sisältyy ja miten erilaiset elinkaarimallit soveltuvat tässä opinnäytetyössä luotavan järjestelmän kehittämiseen. Työn pääpaino oli pistelaskujärjestelmän kehittämisessä, mikä piti sisällään järjestelmän mallintamisen, tietokantarakenteen luomisen ja kehittämisen, käyttöliittymän koodaamisen ja näiden sisältöjen yhteen liittämisen.

Opinnäytetyössä pyritään siis vastaamaan seuraaviin kysymyksiin:

- Millainen elinkaarimalli soveltuu pienten tietokantapohjaisten ohjelmistojen kehitysprosesseihin?
- Mitä vaiheita kuuluu ohjelmistokehitysprojektiin?

Kysymyksiin haemme vastausta konstruktiivisella tutkimusotteella. Käytännön työ ja yleinen teoria näyttelevät tutkimuksessamme merkittävää osaa. Konstruktiona työssämme toimivat tutkimuksessa selvinneen teorian pohjalta kehittämämme pistelaskujärjestelmä ja tähän liittyvä dokumentaatio.

3 Taustatietoa

Suomen Uimaliitto on suomalaisen uinnin kattojärjestö, johon kuuluu neljä eri uintiurheilun muotoa. Suurimman lajin eli kilpauinnin lisäksi muita muotoja ovat uimahypyt, vesipallo ja taitouinti. (Suomen Uimaliitto 2009 a.)

Opinnäytetyön aihetta valittaessa saimme tietää, että taitouinnin pistelaskujärjestelmä on puutteellinen, minkä takia otimme yhteyttä Suomen Uimaliiton taitouinnin johtoryhmän puheenjohtajaan Ulla Luceniukseen. Keskusteltuamme aiheesta selvisi, että liitolla ei tällä hetkellä ole taitouinnin pistelaskujärjestelmää, vaan pisteiden laskeminen toteutetaan manuaalisesti kynän ja paperin avulla sekä käyttäen Microsoftin Excel-taulukkolaskentaohjelmaa. Toinen esiin tulleista ongelmista oli pistelaskun tarkkuuksien yhdenmukaisuus. Eri kilpailuissa käytettiin erilaisia menetelmiä pisteiden laskemiseen ja tulostamiseen, minkä takia samalla suorituksella lopputulokset voivat vaihdella pistelaskun tarkkuudesta riippuen. Lisäksi kyseinen toimintamalli oli erittäin hidas ja hankala, minkä takia uuden järjestelmän kehittäminen todettiin erittäin tarpeelliseksi.

Taitouinti on monipuolinen uintiurheilun muoto, joka vaatii hyvän uimataidon lisäksi kestävyyttä, taitoa ja voimaa. Laji on rantautunut Suomeen 1920-luvulla, mutta nykyisellään lajin harrastajamäärä on varsin pieni verrattuna esimerkiksi kilpauintiin. Lajissa kilpaillaan sekä kuviokilpailuissa että musiikkiohjelmissa. Musiikkiohjelmat pitävät sisällään yksinuinnin eli soolon, pariuinnin eli duon, joukkueen ja näiden yhdistelmän yhdistetyn vapaaohjelman eli combon. (Suomen Uimaliitto 2009 b.)

4 Käsitteistö

Jotta opinnäytetyö olisi lukijalle mahdollisimman helppolukuinen, olisi tärkeää että lukija sisäistäisi aiheeseen liittyvän ydinterminologian. Tämän luvun tarkoituksena on tarjota tarvittava perustietämys, jotta myöhemmin esitettävässä tutkimuksessa sekä projektin kuvauksessa ei sisältö jäisi epäselvälsi aiheeseen perehtymättömälle. Käsitteet on pyritty avaamaan mahdollisimman selkeästi ja tarpeeksi laajasti kuitenkaan paneutumatta tarkoituksettomiin yksittäisiin faktoihin.

Pistelaskujärjestelmä tarkoittaa tietokoneohjelmistoa, jonka avulla voidaan hallinnoida osallistujaluetteloita ja merkitä tuomareiden antamat pisteet. Ohjelma mahdollistaa pisteiden tallentamisen tietokantaan, josta kuvioiden pisteet on helppo noutaa lopputulosten laskemista varten.

Tämän hetkisten sääntöjen mukaan taitouintikilpailuissa tulisi olla 6 tai 7 tuomaria, mutta rakennetussa järjestelmässä on huomioitu myös poikkeustapaukset, joissa tuomareita on vain 3, 4 tai 5. Jokainen tuomari antaa kunkin uimarin jokaiselle arvosteltavalle liikkeelle eli kuviolle pisteet nollasta kymmeneen. Tulos annetaan yhden desimaalin tarkkuudella. Kuvioita uidaan aina yhteensä neljä kappaletta. Lopullisten pisteiden laskeminen kuviokilpailussa tapahtuu seuraavasti: Yksittäisen kuvion pisteistä poistetaan suurin ja pienin arvo, lopuista lasketaan keskiarvo ja tulos kerrotaan kuvion vaikeuskertoimella. Kuvioiden pisteiden summa jaetaan kuvioiden vaikeuskertoimien summalla ja kerrotaan kymmenellä. Lopuksi tulosta vähennetään mahdolliset virhepisteet.

4.1 Käyttöliittymätason käsitteistö

GUI (**G**raphical **u**ser **i**nterface) tarkoittaa graafista käyttöliittymää eli ohjelmiston ruudulla näkyvää osaa, kuten esimerkiksi työpöydällä toimivaa ohjelmaa. Tutkija Jeremy Reimer pitää ikoneita, ikkunoita ja valikoita graafisen käyttöliittymän tärkeimpinä elementteinä. Käyttöjärjestelmässä toimiminen tapahtuu pääasiassa käyttämällä hiirtä eikä kirjoittamalla komentoja näppäimistöllä, kuten ennen kuin graafisia käyttöliittymiä oli olemassa. (Ars Technica 2005.)

HTML-lyhenne tulee sanoista **H**yper**T**ext **M**arkup **L**anguage. Sen juuret ovat lähtöisin SGML -kielestä, joka on laaja asiakirjojen hallintajärjestelmä. HTML:n perusominaisuuksiin kuuluu asiakirjojen rakenteen kuvaaminen, millä ei ole tarkoitus muotoilla Internet-sivun todellista ulkoasua. HTML-kielen tärkeimpiä elementtejä ovat: otsikot, kappaleet, luettelot ja taulukot. (Lemay 1998, 28.)

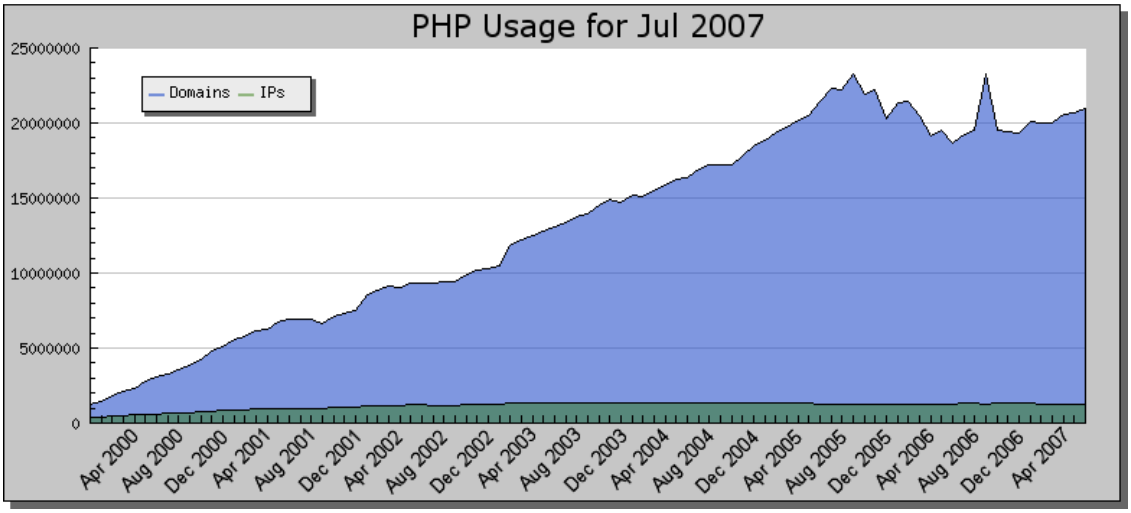
CGI (The **C**ommon **G**ateway **I**nterface) on standardi, joka määrittää selaimen ja palvelimen välillä toimivan ohjelman toimintaa. CGI-ohjelmistot mahdollistavat dynaamisen informaation käsittelyn, mikä antaa uusia mahdollisuuksia HTML-kieleen verrattuna. HTML-kielellä ei pysty tuottamaan uutta tietoa, vaan kuvantamiskieli tulostaa aina jo aiemmin kirjoitetun informaation. CGI tukee useita ohjelmointikieliä, minkä takia se on erittäin yleisesti käytössä oleva standardi. CGI toimii aina tietystä kansiosta, jotta palvelin osaa ajaa ohjelman ennen kuin haluttu tieto tulostetaan selaimeen. Oletuskansiona käytetään yleensä kansiota nimeltä cgi-bin. (NCSA 2009.)

4.2 Sovellustason käsitteistöä

PHP on ohjelmointikieli, jonka kehitti Rasmus Lerdorf vuonna 1995. PHP-lyhenne tarkoitti alun perin sanoja Personal Home Page, mutta vuonna 1997 lyhenteen tarkoitus muutettiin vastaamaan Hypertext Preprocessoria. (Gilmore 2005, 1-2). PHP-ohjelmointikieli on suunniteltu erityisesti Internet-sivuja silmällä pitäen, mutta sitä voidaan soveltaa yleisesti muussakin ohjelmistokehityksessä. Kielen vahvuuksia ovat muun muassa monitoimisuus ja samankaltaisuus moniin muihin ohjelmointikieliin. Tämän lisäksi PHP:lle on olemassa lukuisia laajennuksia, joista tähän opinnäytetyöhön soveltuvimpana esimerkkinä kuuluu MySQL-tietokantojen käsittelyyn liittyvä laajennus. (Heinisuo & Rauta 2007, 26.)

Gilmoren mukaan PHP:n neljä peruspilaria ovat käytännöllisyys, teho, mahdollisuudet ja hinta. Toimivaan PHP-skriptiin ei välttämättä tarvita kuin yksi rivi, ja verrattuna moniin muihin ohjelmointikieliin PHP-kieleen ei tarvitse kutsua erillisiä kirjastoja. PHP helpottaa ohjelmoijan kehitystyötä, sillä kieli ei vaadi muuttujille erillisiä määrityksiä. Muuttujilla tarkoitetaan

ohjelmoinnissa käytettyä tietolokeroa, jonka sisältö voidaan määrittää tarpeiden mukaisesti. Kieli pyrkii selvittämään muuttujien tyypit automaattisesti kun niitä kutsutaan koodissa. PHP-kehittäjä on harvoin sidottu yhteen tiettyyn toteutusratkaisuun, koska PHP:ssä on sisäänrakennettuna tuki yli 25 tietokantatuotteelle. Lisäksi etuna on ohjelmiston toimivuus ilman käyttö-, muokkaus- ja jakelurajoituksia. (Gilmore 2005, 5-8.) PHP:n käyttö on kasvanut tasaisesti lähes koko 2000-luvun, kuten alla olevasta kuvasta voidaan havaita.



Kuva 1: PHP:n käytön kehitys.

Xampp on avoimen lähdekoodin ohjelmistopaketti, joka sisältää kaiken tarpeellisen web-pohjaisen kehitysympäristön luomiseksi. Paketti sisältää muun muassa Apachen web-palvelimen, PHP:n, MySQL:n ja Perlin vaatimat lisäosat. Xampp toimii usealla eri käyttöjärjestelmällä. Xampp:n tukemista käyttöjärjestelmistä merkittävimmät ovat MS Windows, Linux, Solaris ja Mac OS X. Xampp-ohjelmistopakettia käytetään ohjelmistokehitysprojekteissa alustana helpon ja kattavan sisältönsä ansiosta. Ohjelmistopaketin avulla käyttäjä voi rakentaa ja kehittää web-palvelun sisältöä olematta yhteydessä verkkoon. Xampp luo käytettävälle koneelle paikallisen palvelimen, jonka avulla kone käsittelee tietoja. (Xampp 2010.)

Apache HTTP Server on avoimen lähdekoodin ohjelma, jonka tarkoitus on toimia palvelimena niin tavallisilla palvelinasemilla kuin esimerkiksi tavallisilla kotikoneillakin. Lisäksi sen käyttäminen on mahdollista lähes kaikilla käyttöjärjestelmillä. Apache HTTP Server jakaa staattisia tiedostoja HTTP:n avulla, mutta siihen on saatavilla lisäksi erilaisia moduuleja, joiden avulla käyttökohteita tulee paljon lisää. Eri moduulien yhdistelyn mahdollisuuden myötä Apachesta onkin tullut kaikkein suosituin palvelinohjelma. Sen osuus kaikista palvelimista on noin puolet.

4.3 Tietokantatason käsitteistöä

Tietokanta on tietovarasto, johon tallennetaan tietoa eli dataa myöhäisempää käyttöä varten. Tietokannaksi voidaan lukea esimerkiksi erilaiset luettelot, rekisterit ja hinnastot. (Polvinen 1999, 2).

Tässä opinnäytetyössä käytetään SQL-relaatiotietokantaa pistelaskujärjestelmän taustalla, mikä koostuu kaksiulotteisista tauluista. Yksittäinen tietokannan taulu rakentuu tietokentistä eli sarakkeista ja riveistä eli tietueista. Sarakkeisiin tallennetaan aina tietyn tyyppistä tietoa, kuten esimerkiksi merkkitietoa, päiväystietoa ja loogista tilaa (tosi/epätosi). Yhdelle riville tallennetaan aina yhden kokonaisuuden tiedot sarakkeittain. Relaatiotietokantojen vahvuutena on perusavaimien käyttö. Tietokannoissa jokin sarakkeista määritellään perusavaimeksi, jolloin kyseisen sarakkeen tietueet ovat uniikkeja. Täten jokaiselle riville saadaan jokin muista riveistä erottava tietoavain, jonka avulla voidaan yhdistellä tietokenttiä eri taulujen välillä. (Polvinen 1999, 2-3.)

MySQL on suomalaisen Michael ”Monty” Wideniuksen kehittämä relaatiotietokantajärjestelmä, joka pohjautuu SQL-ohjelmointikieleen. MySQL-kielessä on kuitenkin niin paljon eroavaisuuksia verrattuna SQL-kieleen, että voidaan puhua itsenäisestä kielestä. (MikroPC 2004, 54). MySQL:n vahvuuksia ovat sekä avoin lähdekoodi että järjestelmän nopeus ja vakaus. MySQL on maailman suosituin avoimen lähdekoodin tietokantajärjestelmä. (MySQL –Kotisivut, 2010.)

5 Elinkaarimallit

Tässä osiossa käydään läpi yleisellä tasolla tietojärjestelmän elinkaarimallit. Luvussa selitetään lisäksi viisi yleisimmin käytössä olevaa tietojärjestelmän elinkaarimallia sekä niiden ominaispiirteet. Luvussa kahdeksan tarkastellaan tarkemmin elinkaarimallien soveltuvuutta kehitettävälle pistelaskujärjestelmälle.

Tietojärjestelmän elinkaari sisältää kaiken järjestelmän kehittämisen aloittamisesta aina sen käytöstä poistamiseen asti. Elinkaarimalleja sovellettaessa on muistettava, etteivät ne anna yksityiskohtaisia ohjeita järjestelmän rakentamiseen, vaan ohjeet ovat ainoastaan viitteellisiä. (Pohjonen 2002, 39.)

Elinkaarimallit voidaan jakaa pienempiin kokonaisuuksiin, joista yleisimmin käytetty jako on seuraavanlainen: kartoitus, määritys, suunnittelu, toteutus, käyttöönotto ja ylläpito. Jokainen vaiheista sisältää oman käsitteistönsä, työtapansa ja työkalunsa. (Laine 2002, 4.)

Kartoitusvaiheen tarkoitus on määrittää järjestelmän tarkoitus ja kehittämisen kannattavuus. Tässä vaiheessa kartoitetaan erilaisia vaihtoehtoja kuten, pitääkö järjestelmä ostaa kolmannelta osapuolelta tai onko mahdollista teettää yksilöllisiä tarpeita vastaava kokonaisuus. Kartoitusvaiheen aikaiset, karkeat tavoitteet tarkentuvat myöhempien osavaiheiden aikana. (Laine 2002, 6.)

Kartoitusvaiheen jälkeinen määritysvaihe sisältää yksityiskohtaisen suunnitelman järjestelmän ominaisuuksista. Järjestelmän tavoitteet asetetaan joko käyttäjien tai muiden sidosryhmien tarpeiden mukaisesti. Tyypillisiä tavoitteita ovat muun muassa parempi suorituskyky, tehokkuuden parantaminen sekä nykyjärjestelmän ongelmien ratkaiseminen. Tavoitteet tulisi asettaa selkeästi tärkeysjärjestykseen, ja jos mahdollista selkeinä numeraalisina yksiköinä. (Laine 2002, 6-7.)

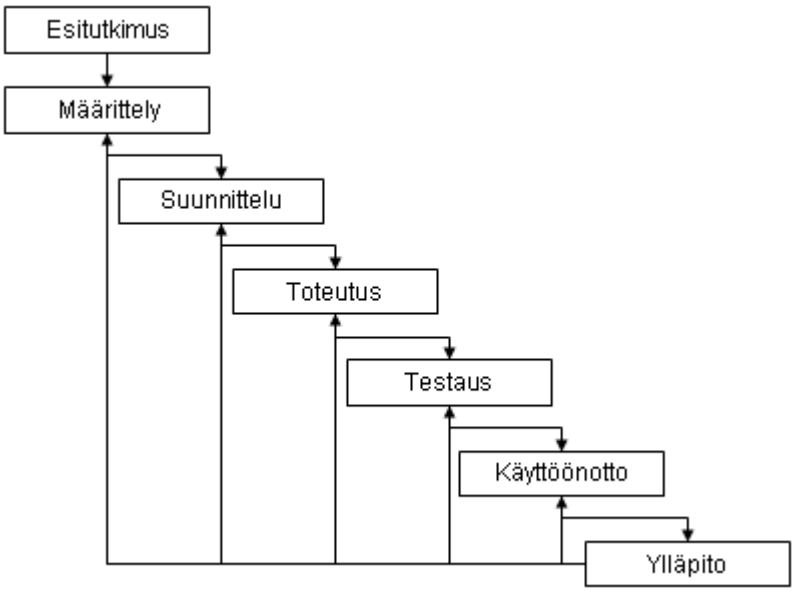
Elinkaarimallin kolmantena osa-alueena on suunnitteluvaihe, joka sisältää kartoitusvaiheen karkean suunnitelman tarkennukset. Siinä myös suunnitellaan miten järjestelmä toteutetaan teknisellä tasolla. Harri Laineen mukaan ”suunnittelun osa-alueita ovat käyttöliittymän suunnittelu, tietokannan suunnittelu, ohjelmistosuunnittelu, laitteiston valinta ja työnkulun suunnittelu” (2002, 8).

Toteutusvaiheessa toteutetaan ja testataan aikaisemmissa vaiheissa määritellyt tietojärjestelmän tarvitsemat ohjelmistot. Tietojärjestelmän ohjelmistojen manuaalit, raportointi ja muu vaadittava materiaali laaditaan heti testauksen jälkeen. Toteutuneiden ohjelmistojen valmistuttua tiedostojärjestelmä asennetaan toimintaympäristöön, ja tarpeelliset vanhat

tiedot integroidaan uuteen järjestelmään. On ensiarvoisen tärkeää, että järjestelmän käytössä ilmenneet virhetilanteet ja puutteet kirjataan ylös, jolloin viimeisessä eli ylläpitovaiheessa järjestelmän epäkohdat voidaan korjata. (Laine 2002, 9.)

5.1 Vesiputousmalli

Vesiputousmalli eli lineaarinen malli on vanhin elinkaarimalleista, mikä on kehitetty 1960-luvun lopulla. Mallin toiminta perustuu ajatukselle, että järjestelmää kehitetään jatkuvasti eteenpäin. Tästä johtuen on hankalaa tai turhaa palata kehitysvaiheessa aikaisempiin versioihin. Seuraavassa kuvassa on kuvattu vesiputousmallin toimintaperiaate.



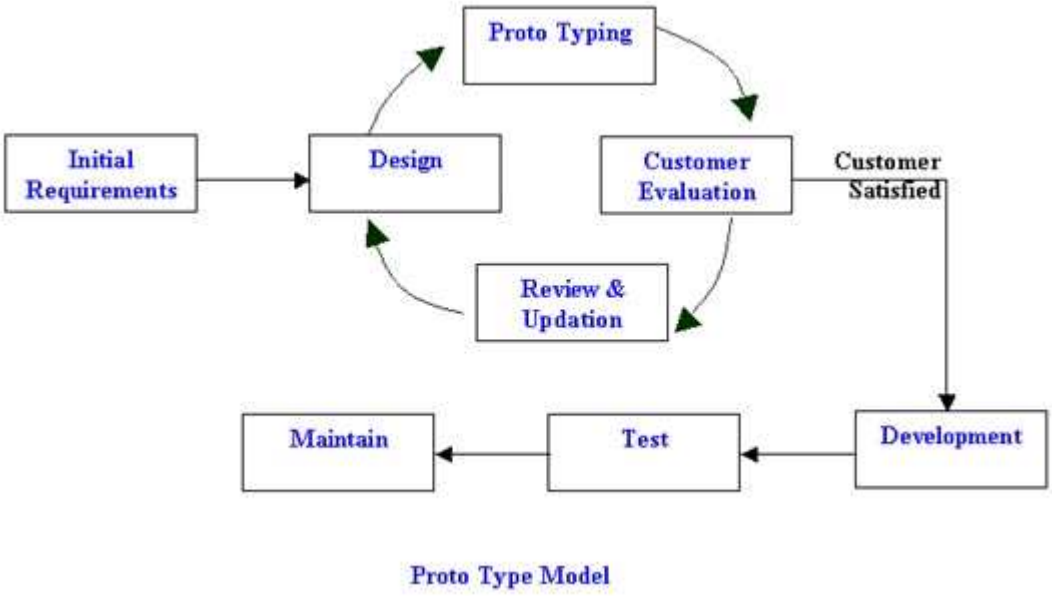
Kuva 2: Vesiputousmalli

Vesiputousmallissa järjestelmän kehittämisen vaiheet käydään läpi vaihe kerrallaan yllä olevan kuvan mukaisesti. Usein oikeassa elämässä kehitysvaiheet etenevät usein limittäin, koska seuraavaan vaiheeseen siirtymisen jälkeen huomataan monesti edellisen vaiheen puutteita, joita täytyy korjata ennen uuden vaiheen aloittamista. Vesiputousmallin merkittävin puute onkin sen iteratiivisuuden kuvaaminen. (Pohjonen 2002, 40.) Lisäongelmia aiheuttaa vaatimusten tiukkuus, sillä malli ei anna tilaa vaatimusten muutoksille. Vesiputousmalli ei ole erityisen asiakaslähtöinen, sillä asiakas näkee tuotteen lopputuloksen varsin myöhäisessä kehitysvaiheessa. Erinäiset ongelmat voivat johtaa aikataulun pettämiseen, joka onkin suurin tekijä vesiputousmallin mukaisesti rakennettujen projektien epäonnistumiseen. (Ohjelmistoprosessit ja ohjelmistojen laatu, 2009, 5.)

Pohjosen (2002, 40) mukaan ”ongelmistaan ja vanhanaikaisuudestaan huolimatta vesiputousmalli on kuitenkin tunnetuin ja edelleen käytännössä yleisimmin sovellettu elinkaarimalli”.

5.2 Prototyyppilähestymistapa

Prototyyppilähestymistapa tarkoittaa elinkaarimallia, jossa asiakkaalle tuotetaan aluksi prototyyppi tilatusta ohjelmistosta, joka sisältää tärkeimmät toiminnallisuudet. Prosessin ensimmäisessä vaiheessa analysoidaan asiakkaan asettamat vaatimukset ja tavoitteet, joiden pohjalta rakennettu prototyyppi annetaan asiakkaalle testattavaksi. Saadun palautteen perusteella ohjelmistoa kehitetään edelleen asiakkaan toiveiden mukaisesti. Edellä mainittua sykliä toistetaan kunnes asiakas on tyytyväinen.

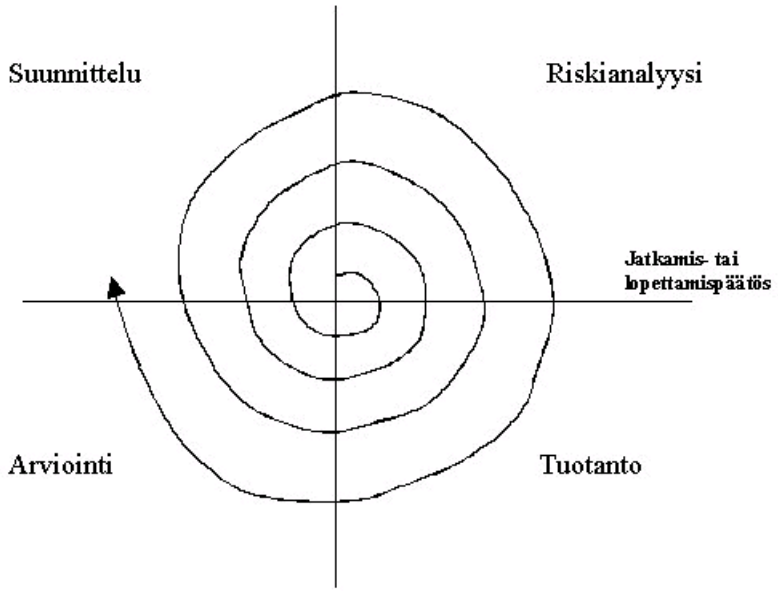


Kuva 3: Prototyyppimalli

Prototyyppimallissa on puutteensa: merkittävimpänä ongelmana on niin sanottu järjestelmän kaksinkertainen rakentaminen. Tästä syystä toimintamalli vaatii paljon resursseja, ja kehitetystä järjestelmästä ei aina saada tuotua esiin kaikkia ongelmia, koska prototyyppi on ainoastaan pelkistetty mallikappale ohjelmistosta. (Pohjonen 2002, 42.)

5.3 Spiraalimalli

Tämän mallin keskeinen ajatus on iteratiivisuus eli suunnittelun ja toteutuksen tekeminen pienissä osissa prosessia toistaen. Spiraalimalli on prototyyppimalliin ja systemaattisen vesiputousmallin yhdistelmä. Mallissa yleisimmin näkyvimpinä nyansseina ovat jatkuva riskien analysoiminen ja projektin uudelleen ohjaaminen riskianalyysin tulosten perusteella. Spiraalimalli koostuu neljästä päävaiheesta, joita ovat suunnittelu, riskianalyysi, tuotanto ja asiakkaan suorittama arviointi. Näitä tarkennetaan niin kauan kunnes järjestelmä on valmis. (Pohjonen 2002, 42.)

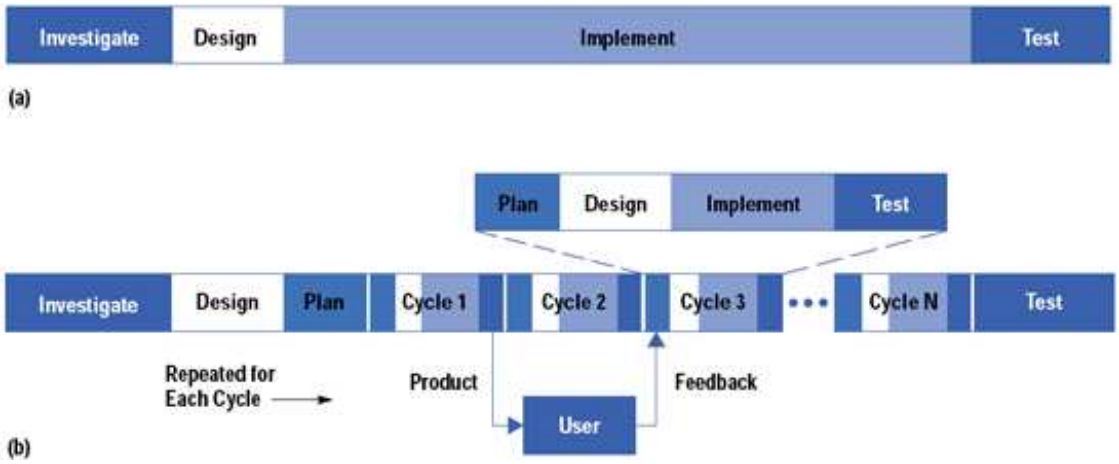


Kuva 4: Spiraalimalli

Spiraalimalli antaa mahdollisuuden soveltaa kehitysvaiheessa muita elinkaarimalleja keskenään. Järjestelmän yksityiskohdat tarkentuvat jatkuvasti projektin edetessä. Yllä olevassa kuvassa havainnollistetaan kuinka yksityiskohdat tarkentuvat iteraation myötä milloin kuvaaja siirtyy ulommaksi keskipisteestä eli lähtökohdasta. Mikäli projektin aikaiset riskit osoittautuvat jossakin vaiheessa liian suuriksi, voidaan projekti keskeyttää. Projektin riskejä pyritään minimoimaan maksimoimalla iteraatiokierroksien määrä. (Pohjonen 2002, 43.)

5.4 Evo-malli

Evo-malli jakaa perinteisen vesiputousmallin kehityssyklin pienempiin osiin, mikä mahdollistaa sen, että projektin tilaajat pääsevät vaikuttamaan tuotteen suunnitteluun jokaisen iteraation aikana. Tämä mahdollistaa aktiivisen palautteen antamisen, mikä puolestaan tehostaa seuraavan iteraation suunnittelua ja kehitystä. Syklin kesto vaihtelee tyypillisesti kahdesta viikosta kuukauteen, ja iterointi jatkuu kunnes projekti on valmis. (May & Zimmer 1996, 1.)



Kuva 5: EVO-malli – Perinteinen vesiputousmalli (a) verrattuna Evo-malliin (b)

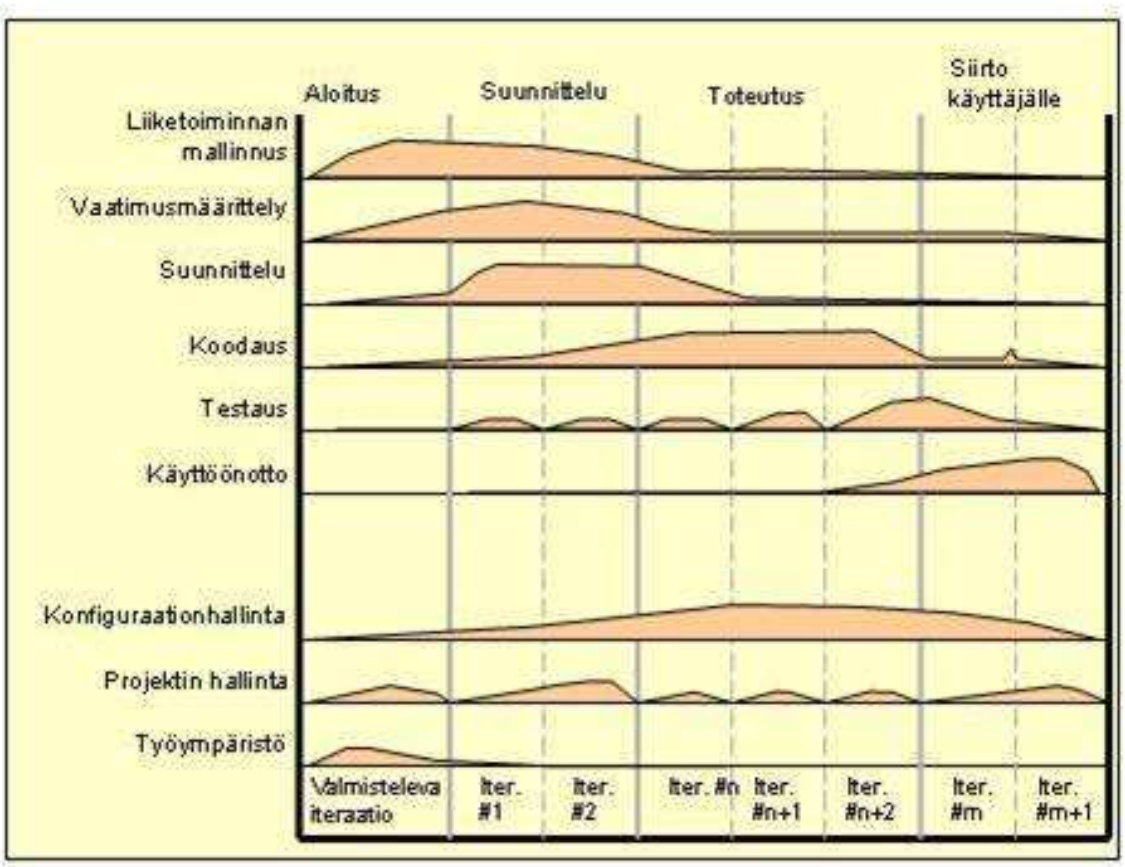
EVO-mallia käyttämällä voidaan vähentää merkittävästi ohjelmistokehityksen riskejä, koska projekti on jaettu pienempiin ja helpommin hallittaviin osiin. Tämä mahdollistaa virheiden tarkemman havaitsemisen, minkä takia ongelmakohtiin voidaan puuttua ajoissa, eikä merkittäviä riskejä tai puutteita synny niin helposti kuin esimerkiksi vesiputousmallissa. EVO-mallin useista iteraatiokierroksista kertyy paljon palautetta, jota voidaan käyttää hyväksi ohjelmiston edelleen kehittämisessä. (May & Zimmer 1996, 1-2.)

5.5 RUP

RUP (Rational Unified Process) on Rational Softwaren kaupallinen ohjelmistokehitysmalli. Malli korostaa iteratiivisuutta, vaatimusten hallintaa, komponenttipohjaista arkkitehtuurirakennetta, visuaalista UML-mallinnusta, laadun valvontaa ja muutosten hallintaa.

RUP:n toiminta-ajatus perustuu neljään eri vaiheeseen, jotka ovat aloitus-, suunnittelu-, toteutus- sekä siirtymävaihe. Näitä osia ei käytännössä toteuteta yhtä kerrallaan, vaan niiden läpivienti on osittain samanaikaista. Yhdenaikaisuus mahdollistaa eri vaiheissa luotujen dokumenttien jatkuvan päivityksen, milloin ne ovat koko ajan kaikkien asianomaisten saatavissa. Aloitusvaiheessa (inception) määritellään kehitysprojektin perusrakenne, jossa määritetään järjestelmän tärkeimmät ominaisuudet, sovellusarkkitehtuuri, projektisuunnitelma ja

mahdolliset riskit. Suunnitteluvaiheessa (elaboration) tarkastellaan ja tarkennetaan aloitusvaiheessa tehtyjä päätöksiä sekä rakennetaan ja dokumentoidaan kattava ohjelmistoarkkitehtuuri. Suunnitteluvaiheelle on ominaista lukuisat iteraatiokierrokset, joiden määrä riippuu halutusta tarkkuudesta ja projektin koosta. Suunnitteluvaihetta pidetään usein RUP-mallin kriittisimpänä vaiheena, koska sen aikana luodaan projektin kivijalka ja varmistetaan kaiken olevan kunnossa ennen seuraavaan vaiheeseen siirtymistä. Ennen siirtymistä seuraavaan vaiheeseen on vielä mahdollista ennustaa kustannuksia ja aikataulua projektin loppuun viemisen osalta. Rakennusvaiheessa (construction) painopiste painottuu projektin viimeisten työvaiheiden ja ohjelmiston toiminnallisuuksien ohjelmointiin ja integrointiin. Kaikki järjestelmän toiminnot testataan yhdessä ja erikseen. Rakennusvaiheen lopputuotoksena syntyy luovutusvalmis tuote. Siirtymävaiheessa (transition) valmis järjestelmä toimitetaan loppukäyttäjien käyttöön. Tarvittaessa tuotteessa ilmenneet ongelmat voidaan korjata ja tuotetta parannella. (Rational 2001, 1-7.)



Kuva 6: RUP –mallin vaiheiden limittyminen

6 Tietojärjestelmän kehittämisen vaiheet

Tietojärjestelmän kehittämisen perustana on usein yrityksen tai järjestön tarve kehittää toimintaansa entistä tehokkaammaksi uusien teknisten mahdollisuuksien tai ympäristön asettamien paineiden myötä. Kehitystyö onkin erittäin systemaattista toimintaa, jossa erilliset toisistaan riippuvaiset osiot seuraavat toinen toistaan loogisessa järjestyksessä. Nämä yksittäiset tietojärjestelmän osaset muodostavat yhdessä järjestelmäkehityksen jo aiemmin kappaleessa viisi mainitun elinkaaren. Järjestelmän kehittäminen pienissä osissa luo perustan aktiiviselle valvonnalle ja etenemisen tarkastelulle prosessin aikana, mikä mahdollistaa muutosten paremman hallinnan. (Pohjonen 2002, 26.)

Nykyään järjestelmän kehittämiseen osallistuvat myös muut kuin IT-alan osaajat. Lähes kaikilla aloilla ja useissa erilaisissa tehtävissä työvälineenä käytetään erilaisia tietojärjestelmiä, minkä takia jo kehitysvaiheessa tulisi olla erilaisia tahoja, jotka edustavat oman alansa näkökulmaa. Erilaiset näkökulmat aiheuttavat usein sekaannuksia, jotka johtuvat Koskimiehen ja Mikkosen (2005, 18) mukaan erilaista kokemusmaailmoista. Sekaannuksista johtuen käyttäjiä ajatellaankin nykyään asiakkaina, minkä takia käyttäjien tarpeiden kuuntelu ja huomioon ottamista pidetään erityisen suuressa arvossa. (Pohjonen 2002, 46.)

Tietojärjestelmän kehittämiseen osallistuvat jaotellaan usein kolmeen kategoriaan: määrittelijät, suunnittelijat ja ohjelmoijat. Määrittelijät huolehtivat esitutkimuksesta, vaatimusmäärittelystä ja järjestelmäanalyysien tekemisestä. Suunnittelijat tekevät määritelmien mukaiset yksityiskohtaiset toteutussuunnitelmat, kuten mallintamisen. Ohjelmoijien tehtäväksi jää laatia määrittelijöiden ja suunnittelijoiden dokumentoinnin perusteella mahdollisimman tarkka käytännön toteutus. Riippuen järjestelmäkehitysprojektin koosta kehittäjät saattavat jakaa työtehtäviä edeltävästä jaksotuksesta poikkeavasti. Loppukäyttäjät toimivat kehittäjien tärkeimpinä apuvälineinä järjestelmän kehityksessä. Käyttäjiltä saadun kohdealuetuntemuksen perusteella saadut tiedot ovat kriittisiä toimivan lopputuloksen saavuttamiseksi. (Pohjonen 2002, 46–47.)

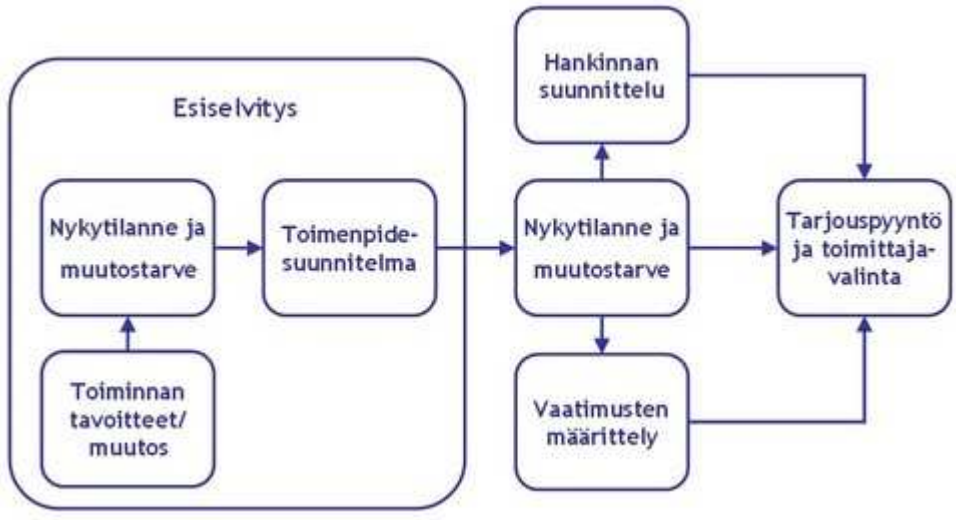
Seuraavissa kappaleissa tarkastellaan yksityiskohtaisesti järjestelmän kehittämisprojektin vaiheita. Käsitteiden perusteellinen ymmärtäminen mahdollistaa järjestelmän rationaalisen kehittämisen minimoimalla riskit ja iterointikierrokset.

6.1 Esitutkimus

Kun päätös järjestelmän kehittämisestä on tehty, edessä on projektin alustavien edellytyksien selvittäminen eli esitutkimus. Esitutkimusvaiheessa selvitetään projektin tavoitteet ja visioidaan mahdolliset suunnitelmat hyvän lopputuloksen saavuttamiseksi. Esitutkimuksen tarkoituksena on tuottaa tietoa järjestelmän kehittämisestä päättävälle taholle ja määrittää pohjatiedot projektin mahdolliselle toteutukselle. Esitutkimus ei vielä tarkoita sitä, että projekti tullaan viemään loppuun asti. Se on vain työkalu, jonka avulla voidaan mitata onko järjestelmää mielekästä ja tarkoituksenmukaista alkaa kehittää. Jos esitutkimus johtaa projektin jatkamiseen, toimii se myös järjestelmän suunnittelun perustana. (Pohjonen 2002, 27.)

Pohjosen (2002, 27) mukaan hyvästä esitutkimusraportista tulisi löytyä seuraavat asiat:

- ”Organisaation tietojenkäsittelyn nykytilanteen kuvaaminen sitä osin kuin se liittyy käsillä olevaan kehityshankkeeseen”
- ”Niiden ongelmien kuvaukset, joihin järjestelmän oletetaan tuovan ratkaisut”
- ”Kuvaukset niistä viite- ja sidosryhmistä, joita hanke koskee”
- ”Alustavien järjestelmälle asetettavien tavoitteiden ja rajausten määritykset”
- ”Uuden järjestelmän kehittämistavoitteiden määritykset”
- ”Eri toimintavaihtoehtojen kuvaukset arvioineen ja perusteluineen”
- ”Alustava suunnitelma tietojärjestelmän kehittämishankkeen läpiviemiseksi”



Kuva 7: Esitutkimuksen (esiselvitys) kulku

6.2 Vaatimusmäärittely

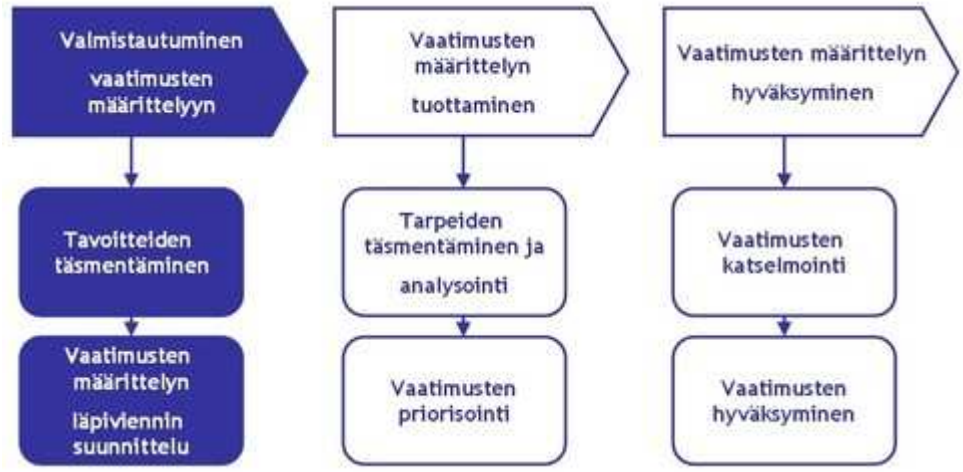
Vaatimusmäärittelyn tarkoituksena on koota yhteen dokumenttiin järjestelmän erilaiset tarpeet ja rajoitukset, että järjestelmä voidaan rakentaa vastaamaan sille asetettuja odotuksia mahdollisimman tehokkaasti. Järjestelmän vaatimukset voidaan jakaa kuuteen ryhmään: asiakkaan vaatimukset, toiminnalliset vaatimukset, laadulliset vaatimukset, suunnittelun vaatimukset, jatkokehityksen asettamat vaatimukset sekä yleiset vaatimukset. (DAU Press 2001, 35–36.)

Hyvässä vaatimusmäärittelyssä tulisi olla ainakin seuraavat elementit: toimeksianto, nykytilan yleiskuvaus, kehitettävälle järjestelmälle asetetut tavoitteet, kaikki järjestelmälle asetetut vaatimukset ja rajoitteet numeroituina ja priorisoituna sekä mahdolliset lisäselvitykset. (Pohjonen 2002, 31).

Pohjosen (2002, 28) mukaan asiakastarpeiden kokoamisen tekee haasteelliseksi standardoidun vaatimusten keräysmenetelmän puuttuminen. Tietoa kerätään monin eri tavoin, kuten esimerkiksi asiakkaan haastattelusta, markkinatutkimuksista ja teknologian tarjoamista mahdollisuuksista. Vaatimusmenettelyn avulla voidaan kartoittaa tarkasti mitä asiakas oikeasti haluaa. (DAU Press 2001, 42.)

Huonosti laaditun vaatimusmäärittelyn takia aikaa saattaa kulua hukkaan, koska silloin täytyy selvittää, mitä aikaisemmin on todella haluttu, ja sen myötä joudutaan mahdollisesti palaamaan vaatimusmäärittelyn alkuun. Suurimmat ongelmat keskittyvät vaatimusten ristiriitaisuuteen ja keskeneräisyyteen. Myös tulkinnaltaan epämääräiset vaatimukset hankaloittavat jatkokehitystä. Yksittäisten vaatimusten todellinen tarkoitus on tärkeää tähdentää dokumenttiin, jotta mahdollisissa epäselvissä tai ristiriitaisissa tilanteissa voidaan luovia kehitystyössä eteenpäin joutumatta palaamaan määrittelyvaiheeseen. Vaatimusmäärittelyä tehdessä tulisi ymmärtää, että rakennettava järjestelmä ei voi kuitenkaan ratkaista ongelmia, joiden syytä ei tiedetä. Jos järjestelmällä pyritään korjaamaan ratkaisematonta ongelmaa, tulee määrittelystä helposti epämääräinen ja vaikeasti ymmärrettävä. (Pohjonen 2002, 30.)

Vaatimusmäärittelyssä kerätty tieto on syytä dokumentoida ja numeroida erikseen, jotta myöhemmin voidaan tarkastella tavoitteita ja niiden toteutumista. Vaatimukset on tarpeellista priorisoida, koska silloin ongelmatilanteissa kehitettäviin järjestelmän komponentteihin voidaan keskittyä tärkeysjärjestyksessä. (Pohjonen 2002, 31.)



Kuva 8: Vaatimusmäärittelyn eri vaiheet

6.3 Kuvausmenetelmät

Tuotteen kehitysprosessin suuntaviivat ja rajoitteet suunnitellaan ohjelmistokehityksessä kuvausmenetelmiä käyttäen. Niiden sisältö skaala vaihtelee toisistaan hyvinkin suuresti riippuen projektin luonteesta sekä toteuttajien toimintatavoista ja mieltymyksistä. Mahdollisina lopputuloksina voi olla erittäin muodollisia ja yksityiskohtaisia tai pelkästään sanallisia hyvin epäformaaleja kuvauksia. Kuvaukset määrittelevät erilaisten tietojen käsittelyssä käytettävät menetelmät, sekä tietojen rakenteen ja esitystavan. (Haikala & Märijärvi 2004, 103.)

6.3.1 Toimintojen ja tietojen kuvaaminen

Algoritmien kuvaustavat voidaan jaotella suorasanaiseen kuvaukseen, pseudokoodiin, vuokaavioon ja ”flowchartiin”. Näistä pseudokoodi tarkoittaa ohjelmointikielen selkokielistä vastinetta, joka ei kuitenkaan ole ohjelmointikoodia, vaan ohjelmointikoodin selkokielinen kuvaus. Yleensä pseudokoodia käytetään moduulisuunnittelussa, jolloin aikaisempi kuvaus muutetaan koodimuotoon. (Haikala & Märijärvi 2004, 104.)

Vuokaavio (tunnettu yleisemmin nimellä kulkukaavio tai activitydiagram) on algoritminen esitysmenetelmä, jossa pseudokoodit kootaan yhdeksi kokonaisuudeksi järjestelmän kokonaiskuvan muodostamiseksi. Todellisessa kehitysympäristössä ei välttämättä tarvita vuokaaviota, sillä sen suunnittelusta saatu hyöty ei vastaa siihen käytettyjä resursseja. Yleensä ohjelmisto suunnitellaan joko suoraan ohjelmointikielellä tai ensin pseudokoodina, joka sitten muutetaan ohjelmaksi. (Haikala & Märijärvi 2004, 104–106.)

6.3.2 Luokkakaaviot

Luokkakaavio on kuvausmenetelmien keskeinen väline, jolla visioidaan tuotettavan ohjelmiston olennaisia käsitteitä ja niiden suhteita toisiinsa. Luokkakaavioihin luetaan oliokaavio, ER-kaavio, tietoyhteyskaavio, käsitekaavio ja kohdekaavio. Näitä käytetään erityisen paljon tietokantasuunnittelussa, koska kaavioita voidaan soveltaa suoraan relaatiotietokannan rakentamisessa. Luokkakaavio on Haikalan ja Märijärven mukaan oliokeskeisten menetelmien tärkein mallinnusväline. (2004, 117–118.)

Luokkakaavioiden tarkoituksena on selventää luokkien välisiä yhteyksiä ja niiden lukumääräsuhteita. Esimerkiksi kahden relaatiotietokannan taulun yhteyttä voidaan tarkentaa lukumääräisesti tai määrittää jonkin taulun riippuvuuksia muihin tauluihin. Kaaviot laaditaan kehittäjän asiantuntemukseen ja asiakkaiden tarpeisiin perustuen, mitä tarkastellaan ja kehitetään edelleen halutun lopputuloksen saavuttamiseksi. (Haikala & Märijärvi 2004, 118, 130.)

6.4 Järjestelmäanalyysi

Järjestelmäanalyysi tarkoittaa Pohjosen mukaan rakennettavan järjestelmän määrittelyä siitä mitä järjestelmän tulee tehdä. Vaatimusmäärittelyä analysoidaan ja sen tuloksista johdetaan toiminnallinen määrittely. Määrittelyvaiheen tärkeimmät osat ovat rajoitteiden tarkentaminen ja kehitettävän järjestelmän looginen kuvaaminen. Järjestelmäanalyysi toimii perustana kaikille kehityksen myöhemmille vaiheille, ja sen pohjalta rakennetaan aikataulut ja töiden jaksottamiset, mikä mahdollistaa esimerkiksi järjestelmän rakentamisen ja testaamisen yhtäaikaisesti. (Pohjonen 2002, 32.)

6.5 Suunnittelu

Esitutkimuksen, vaatimusmäärittelyn ja analyysivaiheen jälkeen siirrytään suunnitteluvaiheeseen. Suunnitteluvaiheen perimmäinen tarkoitus on selvittää miten järjestelmä toteutetaan. Tuotoksena luodaan järjestelmän tekninen määrittely, joka rakentuu toiminnallisen määrittelyn pohjalta. Teknisessä määrittelyssä järjestelmän toteutus jaetaan yleensä kahteen osaan: arkkitehtuuri- ja moduulisuunnitteluun. Arkkitehtuurisuunnittelu jäsentää järjestelmän rakennetta yleisellä tasolla, ja määrittää järjestelmän sisältämät osakokonaisuudet eli moduulit. Moduulisuunnittelussa suunnitellaan kokonaisuuden yksityiskohtia ja tarkempia toimintoja, jotka käsittävät aina jonkin tietyn osakokonaisuuden toiminnot järjestelmästä. Hyvin visioidussa arkkitehtuurisuunnittelussa moduulit toimivat itsenäisinä kokonaisuuksina. Tämä vähentää ongelmatilanteita ja helpottaa järjestelmän ja yksittäisten moduulien testaamista, kun rakenteellisella suunnittelulla voidaan todentaa ongelman alkuperä.

Arkkitehtuurisuunnittelu pystytään jakamaan vielä pienempiin osakokonaisuuksiin, joita kutsutaan alimoduuleiksi. Suunnitteluvaiheessa yksittäinen moduuli ei saisi sisältää suurta määrää alimoduuleja, jottei siitä tule liian monimutkaista toteuttaa. (Pohjonen 2002, 32–33.)

Arkkitehtuurisuunnittelun jälkeen suunnitellaan jokaisen moduulin sisäinen rakenne mahdollisimman tiiviisti, että moduulit olisivat selkeät ja hyvin hallittavissa. Voidaankin todeta, että onnistunut suunnittelu ja ennen kaikkea arkkitehtuurisuunnittelu johtaa toimivaan moduulijakoon ja helpommin toteutettavissa sekä hallittavissa olevaan tietojärjestelmään. (Pohjonen 2002, 33–34.)

6.6 Toteutus

Toteutusvaiheessa tuotetaan alustavasti asetettujen vaatimusten ja toiminnallisen sekä teknisen määrittelyn mukainen järjestelmä. Jotta tavoitteet voitaisiin saavuttaa, tulisi Pohjosen (2002, 34) mukaan toteutusvälineen valinnassa huomioida: sovellusalue, käytetyt menetelmät ja ohjelmistotuotannon mallit, tehokkuusvaatimukset ja toteutus- ja käyttöympäristö. Sovellusalueella tarkoitetaan kussakin tilanteessa juuri siihen soveltuvia toteutusvälineitä. Esimerkiksi SQL–kieli soveltuu käytettäväksi relaatiotietokantasovelluksessa, mutta ei välttämättä systeemiohjelmoinnissa. Pohjonen määrittelee toteutusvälineet vastaamaan mahdollisimman tarkasti suunnittelu– ja vaatimusmäärittelyvaiheissa tehtyjä linjauksia.

Toteutusvaiheessa on syytä tarkastella aikaisempien vaiheiden suuntaviivoja, että järjestelmän siirrettävyys ja ylläpidettävyys pysyvät selkeänä ja stabiilina. Ohjelmakoodia kirjoittaessa tulisi muistaa dokumentoida tai kommentoida koodin eri toimintoja ja vaiheita, jotta tarvittaessa kuka tahansa pystyy seuraamaan ohjelmakoodin logiikkaa tai kehittämään sitä edel-

leen. Monesti yksinkertainen ja selkeä koodin jäsentely onkin parempi ratkaisu kuin liiallinen hienostelu. Järjestelmän toteutuksessa on tärkeää ymmärtää suunnitteluprosessia, jotta sitä voidaan seurata loogisesti edeten. Suunnittelijan tulee ymmärtää toteutusvaiheen standardisoidut perusteet, jotta suunnitelma ja toteutus voisivat kohdata toisensa. (Pohjonen 2002, 35.)

6.7 Testaus

Ohjelmistotuotannossa testauksen tarkoitus on virheiden paikantaminen ja ongelmatilanteisiin johtavien syiden selvittäminen. Testaukseen käytetty työpanos lohkaisee ohjelmistokehitysprojektiin varatuista resursseista jopa puolet. Ohjelmistojen testaaminen toteutetaan käyttämällä sattumanvaraisia toimituksia ja syötteitä, minkä avulla voidaan parantaa ohjelman toimivuutta tai paikantaa koodista löytyvät mahdollisia virheitä. Ohjelman testauksessa ei voi huomioida kuin murto-osa mahdollisista tilanteista, minkä takia vähäisimpiä virheitä voi jäädä huomaamatta. Vaikka testit ja testitulosten aikaansaamat korjaukset parantavat ohjelmakoodia, ei ohjelmiston kehittäjän kannata luottaa ohjelman 100-prosenttiseen toimivuuteen. (Haikala & Märijärvi 2004, 283–287.)

Ohjelmistotuotannossa käytettyjä testaustasoja on erilaisia, ja niillä kullakin on oma tarkoituksensa. Moduulitestaus tarkoittaa ohjelmiston yksittäisen moduulin eli toiminnallisuuden testaamista. Yleensä yksittäinen moduuli koostuu maksimissaan 1500 ohjelmarivistä, ja sen testaamisen suorittaa moduulin toteuttaja. Moduulin toimintaa verrataan yleensä määrittelydokumenttiin, että voidaan varmistua halutuista toiminnallisuuksista. Kun yksittäisiä moduuleita on useampia, ja näiden toimintaa on testattu yksittäisinä toimintoina, voidaan aloittaa integrointitestaus. Integrointitestauksessa sovitetaan yksittäisten moduulien rajapinnat toisiinsa sopiviksi, että niitä voitaisiin liittää yhteen ja suorittaa moduulitestauksen rinnalla. Kun kaikki moduulit on koottu yhteen yhdeksi toimivaksi kokonaisuudeksi, voidaan siirtyä integrointitestauksesta järjestelmätestausvaiheeseen, jolloin kehitettävää kokonaisuutta verrataan määrittelydokumentaatioon ja asiakasdokumentaatioon. Järjestelmän testausvaiheessa testaajina tulisi toimia mahdollisimman paljon järjestelmän loppukäyttäjiä. Järjestelmätestausvaiheeseen luetaan mukaan myös kuormitus-, luotettavuus-, asennus- ja käytettävyystestit. Käytettävyystestaus on asiakkaan kannalta tärkeä, että voidaan varmistua asiakkaan asettamien vaatimusten täyttymisestä. Käytettävyystestauksessa testataan pääasiassa käyttöliittymää valvotuissa koetilanteissa, minkä takia mahdolliset epäkohdat voidaan tunnistaa nopeasti muun muassa videoimalla ja haastattelemalla käyttäjiä. (Haikala & Märijärvi 2004, 288–291.)

Moduuli-, integrointi- ja järjestelmätestausvaiheita on hyvä dokumentoida, että voidaan selvittää mitä asioita on huomioitu ja missä mahdollisesti voi olla puutteita. Suurissa ohjelmistokehitysprojekteissa jokaiselle testausvaiheelle tulisi luoda oma testaussuunnitelma, mutta

pienemmissä projekteissa yhteinen testaussuunnitelma voi olla riittävä. Suunnitelmasta tulisi selvitä, minkälaisia testejä suoritetaan, millä aikataululla ja mitä lopputuloksia näillä toimenpiteillä odotetaan saavutettavan. Testaussuunnitelmasta on löydyttävä testauksen lopettamiskriteerien määrittelyt, jonka perusteella kehitystyö lopetetaan, ja ohjelmisto siirretään käyttöönottovaiheeseen. (Haikala & Märijärvi 2004, 299.)

6.8 Käyttöönotto

Käyttöönottoa voidaan ajatella prosessina, joka sisältää kaikki vaiheet vaihtamispäätöksestä käyttöönottovaiheeseen. (Halonen, 2) Käyttöönoton sujuvuuden kannalta on Pohjosen (2002, 37) mukaan huomioitava sekä siirrettävät tiedot, tiedostot ja tietokannat että muut mahdolliset, aikaisemmat ohjelmistoversiot tai rinnakkaiset ohjelmistot. Käyttöönotossa on myös huomioitava uusien käyttäjien kouluttaminen tai ainakin riittävien käyttöohjeistusten laatiminen.

6.9 Jatkokehitys ja ylläpito

Ohjelmistokehitysprosessin viimeinen ja pisin vaihe on ohjelmiston ylläpito ja mahdollinen jatkokehitys. Tämän vaiheen aikana pyritään kiinnittämään huomiota toimintakunnon ylläpitoon paikkaamalla mahdollisia epäkohtia ja muovaamaan jo olemassa olevia toiminnallisuuksia tarpeiden mukaan sekä suorittamalla muita tarpeellisia muutostoimenpiteitä. Ylläpito voidaan jakaa neljään osioon: korjaavaan, sopeuttavaan, täydentävään ja ennakoivaan ylläpitoon. (Pohjonen 2002, 37.)

Ylläpidon kannalta järjestelmän dokumentointi tulisi hoitaa hyvin, koska yleensä suurimmat ongelmat syntyvät puutteellisen dokumentaation vuoksi. Kehitysprosessin vaikeatajuisuuden takia virheiden paikantaminen voi muodostua haasteelliseksi toimenpiteeksi. Kaikista järjestelmään tehdyistä operaatioista tulisi tehdä kirjallisia muistiinpanoja. Ylläpitovaiheen perusta rakennetaan järjestelmän suunnitteluvaiheessa, missä pyritään ottamaan huomioon kaikki järjestelmässä tapahtuvat muutokset jo etukäteen. (Pohjonen 2002, 37-38.)

7 Case: Taitouinnin pistelaskujärjestelmä

Tässä kappaleessa kerromme opinnäytetyön CASE – osuudesta, joka sisältää tärkeimmät tiedot Suomen Uimaliitolle tekemästämme taitouinnin pistelaskujärjestelmästä. Käymme läpi rakentamamme tietojärjestelmän eri prosesseja, joita ovat mm. määrittely, suunnittelu ja toteutus. Tuomme esiin lyhyesti myös läpi perusajatuksen siitä, kuinka idea rakennettavasta järjestelmästä syntyi ja kuinka meidän kohdallamme järjestelmän rakentaminen eteni.

7.1 Projektin vaiheet

Projektin alussa lähestyimme Suomen uimaliittoa saatuamme tiedon mahdollisesta kehitystarpeesta, joka liittyi taitouinnin sen hetkiseen pistelaskujärjestelmään. Totesimme uimaliiton tarjoaman mahdollisuuden mielekkääksi haasteeksi, sillä tämä tarjoaisi tilaisuuden toteuttaa itseämme. Samalla auttaisimme panoksellamme Suomen taitouinnin nykytilaa. Pistelaskujärjestelmän kehittäminen antaisi mahdollisuuden kehittää edelleen aikaisemmin opittuja taitoja niin ohjelmoinnin kuin mallintamisenkin saralla.

Projektia työstäessämme huomasimme käytännön ja teorian poikkeavan toisistaan vaihdellen projektien sisällöstä riippuen. Teoria antaa vain suuntaviivoja, joita ei pidä ottaa kirjaimellisesti projektin kehitysvaiheissa. Pistelaskujärjestelmän kehittämisprojektiin ei ollut tarpeen sisällyttää kaikkea ohjeistuksen mukaista dokumentaatiota, ja jouduimme jättämään osan dokumentaatiosta pois aikataulullisista syistä. Pienissä projektissa dokumentointiin käytetty aika voidaan käyttää huomattavasti tehokkaammin järjestelmän kehittämiseen.

Seuraavissa kappaleissa keskitymme kuvaamaan pistelaskujärjestelmän suunnittelua, kehittämistä ja rakentamista. Tarkastelemme myös lyhyesti esitutkimuksen, ohjelmoinnin, testauksen ja käyttöönoton vaiheita.

7.1.1 Esitutkimus

Alustavan keskustelun jälkeen sovimme tapaamisen Vantaan Martinlaaksoon Suomen Uimaliiton taitouintijaoston puheenjohtajan Ulla Luceniuksen kanssa. Keskustelussa selvitimme uuden ohjelmiston tarpeita, ja meidän mahdollisuuksia täyttää järjestelmän rakentamiseen asetettuja odotuksia.

Ulla Luceniuksen kanssa käydyn keskustelun pohjalta aloitimme esitutkimuksen, joka käsitti niin taitouinnin pisteytyksen perusteita kuin PHP-ohjelmoinnin ja MySQL-tietokannan soveltuvuutta tämän tyyppiseen projektiin. Muodostimme alkuun käsityksen siitä, kuinka suuri työmäärä meillä olisi edessämme, ja riittäisikö sen hetkinen osaaminen projektin läpiviemiseen

mielekkäässä aikataulussa. Käytimme esitutkimukseen reilusti aikaa, että saimme muodostettua realistiset rajat ja suuntaviivat, joiden avulla kehitysprojektia lähdettäisiin viemään eteenpäin. Esitutkimusta ei erikseen dokumentoitu vaan esitimme uimaliiton edustajalle karkean suunnitelman, joka kattoi järjestelmän oleellisimmat toiminnallisuudet ja periaatteet. Alustavien suunnitelmien pohjalta meille rakentui selkeä kuva siitä, minkälainen elinkaarimalli soveltuisi juuri meidän ohjelmistokehitysprosessille. Puntaroituamme vaihtoehtoja päädyimme valitsemaan prototyyppimallin.

7.1.2 Määrittely ja suunnittelu

Projektimme määrittely- ja suunnitteluvaiheet kulkivat yhdenaikaisesti koko projektin ajan. Ensimmäinen vaihe esitutkimuksen jälkeen oli tarkempi palaveri Ulla Luceniuksen kanssa, minkä tarkoituksena oli tarkentaa sekä meille että asiakkaalle pistelaskujärjestelmän tarpeet.

Toisessa vaiheessa lähetimme kysymyslomakkeen ohjelmiston mahdollisille loppukäyttäjille ja muille taitouinnin kilpailuorganisaation toimihenkilöille. Lomakkeen avulla selvitimme mitä ohjelmiston ominaisuuksia tulisi painottaa ja kehittää. Vastauksien pohjalta tarkemmat toiveet ja suuntaviivat selvisivät, mikä auttoi hahmottamaan mitä pistelaskujärjestelmän tulisi sisältää, ja kuinka se voitaisiin rakentaa käytännössä. Konkreettisin esimerkki suunnitteluvaiheen tuotoksista on tietokantarakenteen suunnitteleminen.

Määrittelydokumentit laadittiin haastatteluista ja kysymyslomakkeesta saatujen tietojen perusteella. Käyttötapauskuvausten tarkkuus määriteltiin suppeaksi ja suuntaa antavaksi, koska sekä asiakkaan että meidän tietotekninen osaaminen asetti omat rajoitteensa. Projektiorganisaation pienuus vähensi tarkkuusvaatimuksia, koska kaikki tuotantoon osallistuvat henkilöt olivat mukana jokaisessa yksittäisessä vaiheessa. Suurpiirteiset käyttötapauskuvaukset toimivat suuntaa antavana toimintoluettelona tarjoten suunnittelu- ja toteutusvaiheessa vapaammat kädet. (Vaatimusmäärittely, kysymyslomake, käyttötapauskuvaukset ja järjestelmän toimintoja kuvaavat kaaviot ovat liitteinä tämän opinnäytetyön lopussa.)

7.1.3 Toteutus

Järjestelmän toiminnallisuudet rakennettiin PHP-kielellä lukuun ottamatta muutamaa yksittäistä poikkeusta, jotka toteutettiin JavaScriptillä parantamaan tulosteiden hallintaa. Poikkeustapaukset on yritetty tarkoituksella jättää mahdollisimman vähiin, ettei toiminnallisuus kärsi selainten asettamien rajoitteiden takia. Järjestelmän ohjelmakoodi on sisällytetty html-koodiin, milloin järjestelmä toimii Internet-selaimessa.

Järjestelmä ohjelmoitiin käytettäväksi ensisijaisesti Mozilla Firefox – selaimella, minkä vuoksi muut selaimet eivät välttämättä näytä järjestelmän ulkoasua oikein. Mielestämme selaimista Firefox mahdollistaa käyttöliittymän optimoinnin muita selaimia paremmin. Firefox tukee laajasti haluttuja toiminnallisuuksia kuten JavaScriptiä ja erilaisten muotoilujen mahdollisuus on moninainen. Lisäksi kyseisen selaimen toimivuus on hyvä yleisimmissä käyttöjärjestelmissä. (Se on muun muassa Ubuntun oletusselain.)

Ohjelmointiprosessin edetessä ja taitojen karttuessa koimme mahdolliseksi etsiä erilaisia ja innovatiivisia tapoja toteuttaa ohjelman toiminnallisuuksia PHP–kielellä. Esimerkiksi pisteytystoiminnallisuus on rakennettu osittain käyttäen korkeamman tason tietokantayhteyttä (prepare –statement). Ohjelmointiprosessi oli opinnäytetyön aikaa vievin osa-alue; oman arviomme mukaan kokonaistyömäärästä ohjelmointi on vienyt reilusti yli puolet. Järjestelmään tuottamamme sisältö rakentuu 40 PHP–tiedostosta ja noin 12 000 koodirivistä.

Järjestelmän ensimmäinen prototyyppi sisälsi karkean version urheilijoiden lisäämisestä kilpailuun ja kilpailun pisteytysvaiheen. Prototyyppi ei mahdollistanut tulosteiden luomista, kuten osallistujaluetteloita tai lopputuloksia, vaan näitä tietoja voitiin tarkastella ainoastaan suoraan tietokannasta. Prototyypistä saadun palautteen pohjalta järjestelmää kehitettiin edelleen.

Toteutusvaiheen lopuksi kehitimme ensimmäisen julkaisuversion, joka toimi pistelaskujärjestelmänä kuviokilpailuissa. Versiosta jätettiin tarkoituksella pois musiikkiohjelmien osuus aikataulullisista syistä. Järjestelmän suunnittelussa huomioitiin jatkokehityksen kannalta tärkeä tietokannan rajapinta, kuten valmiiksi suunniteltu kuvio-osuuden pisteiden käyttö musiikkiosuuden pisteytyksessä.

7.1.4 Testaus ja käyttöönotto

Suoritimme testausta koko järjestelmän kehityksen ajan. Alkuvaiheen tärkein testausmenetelmä oli moduulitestaus, jonka avulla saatoimme välittömästi varmistua yksittäisten järjestelmän osien toimivuudesta saatuamme ne valmiiksi. Moduulitestauksella pyrimme paikantamaan virheet ja epäkohdat mahdollisimman aikaisessa vaiheessa. Kun ohjelman ydintoiminnallisuudet saatiin toimimaan yksittäisinä toimintoina, aloitimme integrointitestauksen. Järjestelmää testattiin erikokoisina kokonaisuuksina, milloin pystyimme puuttumaan ongelmatilanteisiin moduulien välisessä kommunikaatiossa, kuten kilpailun perustaminen ja urheilijoiden lisääminen perustettuun kilpailuun tai kilpailun pisteyttäminen ja lopputulosten tarkasteleminen.

Testasimme järjestelmän viansietokykyä aiheuttamalla tahallisesti virhetilanteita ja luomalla näin yllättäviä ongelmatilanteita. Jaoimme testaamamme testitilanteet tarkempiin kokonaisuuksiin: tietokantayhteyksiin ja kilpailijoiden hallinnointiin, kilpailun läpivientiin, tulosteiden hallintaan ja kilpailutestaukseen.

Ohjelman tietokantayhteyttä ja kilpailijoiden hallinnointia testasimme luomalla ohjelmaan kuvitteellisia kilpailijoita, ja tarkastamalla tämän jälkeen tietojen päivittymisen tietokantaan. Kun edellinen testitapahtuma toimi moitteitta, muokkasimme suoraan tietokannasta kyseisen testikilpailijan tietoja, millä voitiin varmistaa tietokantayhteyden toimiminen järjestelmään ilman käyttäjän syöttämiä syötteitä. Lopuksi kilpailija poistettiin järjestelmän käyttöliittymän avulla tietokannasta, ja varmistimme kilpailijan poistumisen sekä tietokannasta tulostuvasta listasta että tietokannasta. Pyrimme saamaan ongelmatilanteita aikaiseksi käyttämällä erikoismerkkejä ja liian pitkiä nimiä, milloin pystyimme toteamaan kuinka järjestelmä reagoi ongelmatilanteissa. Erikoismerkkien ongelmien takia vaihdoimme ohjelmiston PHP-kielen merkistöä käyttötilanteeseen sopivammaksi. Liian pitkien nimien kohdalla päädyimme määrittämään maksimipituuden riittäväksi, milloin virhetilanteen toteutuminen todellisessa käyttötilanteessa muuttui huomattavasti epätodennäköisemmäksi. Suoritimme saman testausprosessin myös tuomareiden ja perustettujen kilpailuiden kohdalla. Kolmannessa merkittävässä testissä lisäsimme urheilijoita johonkin haluttuun sarjaan, minkä jälkeen poistimme sattumanvaraisen uimarin kyseisestä sarjasta. Tällä testasimme osallistujanumeroinnin reagoimista poistettaessa kilpailijoita osallistujaluettelosta. Testasimme onko kilpailun samaan sarjaan mahdollista ilmoittaa yksittäistä uimaria useaan kertaan. Havaittuamme edellä mainitun skenaarion mahdolliseksi, teimme muutoksia järjestelmän toiminnallisuuteen epäkohdan korjaamiseksi.

Kilpailun läpivientiosioissa testasimme kilpailutilanteen pisteytysmoduulia. Perustimme sarjan, johon ilmoitimme joukon kuvitteellisia kilpailijoita. Kilpailijoille annettiin sattumanvarai-

sia pisteitä varmistuaksemme siitä, että annetut pisteet tallentuvat tietokantaan aina oikean lisenssinumeron kohdalle. Tämän vaiheen toisessa testitapauksessa keskityttiin varmistamaan kunkin kilpailijan pisteytyksen toimivuus huolimatta siitä, kuka kilpailijoista aloittaa uitavan kuvion. Testausta suoritettiin erikokoisilla kilpailijamäärillä siten, että jokainen kilpailija aloitti vuorollaan kuvion. Seuraavassa testissä selvitettiin tulostaisiko järjestelmä oikean määrän pisteytysruutuja kilpailuun. Pisteytysruutujen lukumäärän tuli olla sama kuin tuomareiden määrä. Mahdollisia vaihtoehtoja tuomareiden lukumäärästä oli viisi, minkä takia tämä testi suoritettiin kaikilla mahdollisilla tuomarikokoonpanoilla.

Tulosteiden hallinnan testaus sisälsi osallistujaluetteloiden, lopputulosten ulkoasun asettelun ja pisteytyksen toimivuuden testaukset. Koska osallistujaluettelo on pelkkä tuloste tietokannasta, sen tietojen oikeellisuus testattiin vertailemalla nimiä ja osallistujanumeroita keskenään. Pisteytyksen toimivuus testattiin tarkastelemalla kilpailutilannetta ja laskemalla lopputulokset sekä manuaalisesti että järjestelmän avulla. Saatuja tuloksia verrattiin keskenään, milloin pystyimme varmistumaan järjestelmän tulosten paikkansapitävyydestä. Testi suoritettiin eri tuomarimäärillä, jolloin voitiin varmistua pisteytysmallien toimivuudesta. Ulkoasun asettelun testauksessa testasimme kunkin tietueen tulostumista halutulle paikalle. Testissä tarkasteltiin erilaisia kilpailuja ja tulostamalla kustakin lopputulokset. Näytöllä näkyvien tulosten lisäksi testasimme paperitulosteen muotoilun toimivuutta erikokoisilla kilpailutilanteilla. Testillä haluttiin varmistua tulosteiden sivunvaihtojen ja jokaisen sivun alalaitaan tulostuvan alatunnisteen toimivuudesta.

7.2 Kysymyslomake

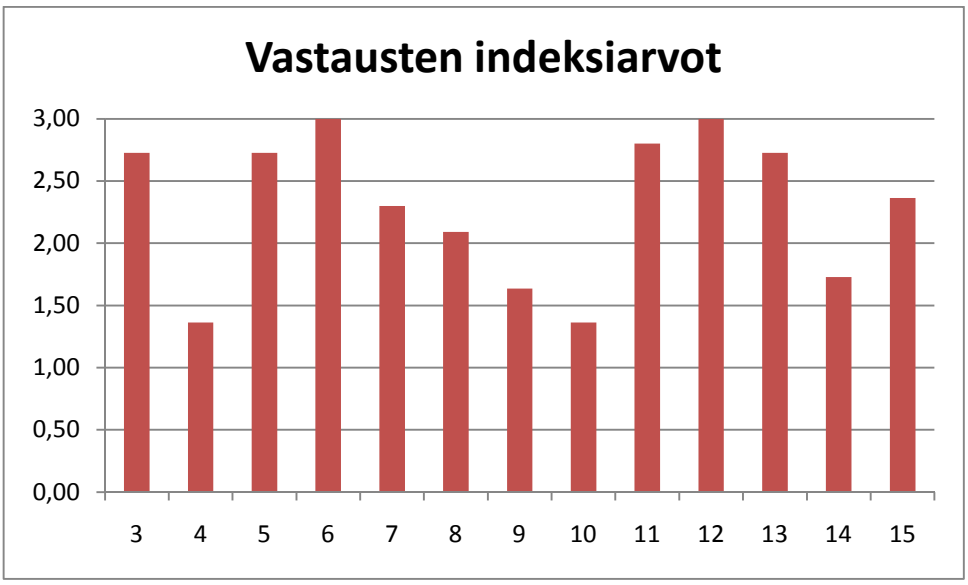
Tässä kappaleessa tarkastelemme kysymyslomakkeen avulla ohjelman loppukäyttäjiltä kerättyä informaatiota. Informaatio sisältää käyttäjien mielipiteitä ohjelman eri toiminnoista ja yleisesti ohjelman tarpeista asteikolla yhdestä neljään.

Kysymyslomakkeen kohdissa yksi ja kaksi vastaajaa pyydettiin antamaan perustietoja itsestään, että eri käyttäjäryhmät voitaisiin erotella toisistaan. Kysymyslomakkeen muissa kohdissa vastaajan tuli valita sopivin neljästä vaihtoehdosta (paljon, melko paljon, vähän ja ei lainkaan). Vastausvaihtoehdot rajattiin tarkoituksella neljään, jolloin vaihtoehtojen määrä oli parillinen. Vastaaja joutui näin ollen arvioimaan vastaustaan tarkemmin sen sijaan, että vastaisi neutraalisti keskimmäisen vaihtoehdon. Seuraavaksi on listattu kysymyslomakkeen kysymykset.

1. Sukupuoli:
2. Ikä:
3. Kuinka paljon tarvetta olisi taitouinnin pistelaskujärjestelmän kehittämiselle?
4. Oletteko käyttänyt aikaisemmin jotakin taitouinnin pistelaskujärjestelmää?
5. Kuinka paljon ohjelman helppokäyttöisyys merkitsee?
6. Kuinka tärkeää on, että järjestelmä laskee uimarien pisteet itsenäisesti?
7. Kuinka tarpeellista on, että uimarien tiedot tallentuvat järjestelmään?
8. Kuinka tarpeellista on , että tuomareiden tiedot tallentuvat järjestelmään?
9. Kuinka tärkeää, että ohjelma on suomenkielinen?
10. Kuinka tärkeää on ohjelman visuaalinen ulkoasu?
11. Kuinka tarpeellista on, että ohjelmaa voi käyttää ilman Internet yhteyttä?
12. Kuinka tärkeää on, että ohjelmasta voi tulostaa lopulliset tulokset?
13. Kuinka tärkeää on, että tuloksissa on eriteltyinä kuviokohtaiset tulokset?
14. Kuinka tärkeää on, että tuloksissa on eriteltyinä tuomarikohtaiset tulokset?
15. Kuinka tärkeää on järjestelmän helppo asennus?

7.3 Kyselyn vastausten analysointi

Lähetimme kysymyslomakkeen 16 henkilölle, joista 11 palautti kysymyslomakkeen takaisin analysoitavaksi. Vastanneista 82 prosenttia oli naisia ja vastanneiden keski-ikä oli noin 33 vuotta. Kuvan pystyakselilla olevat arvot kuvaavat toiminnon tai ominaisuuden tärkeyttä, missä 0 edustaa vähiten tärkeää ja 3 vastaavasti erittäin tärkeää. Vaaka-akselin numerot vastaavat kysymyslomakkeen kysymysten numerointia.



Kuva 9: Kysymyslomakkeen vastausten indeksiarvot.

Kaavion perusteella kyselyyn vastanneiden mielestä ylivoimaisesti tärkeimpiä ominaisuuksia olivat järjestelmän kyky laskea pisteitä automaattisesti ja tulostaa lasketut pisteet tarvittaessa. Vastanneiden mielestä olisi myös tärkeää pystyä erittelemään tuloksissa kuviokohtaiset pisteet. Kyselyn perusteella muita arvostettuja ominaisuuksia olivat helppokäyttöisyys ja mahdollisuus käyttää ohjelmistoa ilman verkkoyhteyttä.

Kriittisimpien toimintojen lisäksi kyselystä selvisi, että olisi hyödyllistä, jos järjestelmään tallentuisi uimareiden ja tuomareiden tietoja myöhäisempää käyttöä varten. Melko tärkeänä pidettiin järjestelmän asennuksen helppoutta. Hieman yllättäen ja aikaisemmista haastatteluista poiketen, ominaisuus, jossa tuloksiin tulostuu tuomarikohtaisesti eritellyt pisteet, ei ollut odotetun tärkeä ominaisuus.

Kyselystä selvisi, että järjestelmän kehittäminen on ehdottomasti tarpeellista, vaikka tähän asti suurin osa kilpailuiden toimitsijoista on pärjännyt ilman erillistä pistelaskujärjestelmää. Järjestelmän ulkoasu ja suomenkielisyys eivät nousseet tärkeysjärjestyksessä kovinkaan korkealle muihin ominaisuuksiin verrattuna.

7.4 PHP & MySQL valinta

Järjestelmä perustuu PHP-ohjelmointikieleen ja MySQL-tietokantaan. Ohjelmointikielen ja tietokantasovelluksen tuli olla peruslähtökohdaltaan avoimeen lähdekoodiin perustuvia ohjelmistoja, jotka ovat vapaasti ladattavissa. Tämän lisäksi koulussa saamamme perusopetuksen pohjalta oli luontevaa valita jokin meille ennalta tuttu ratkaisu.

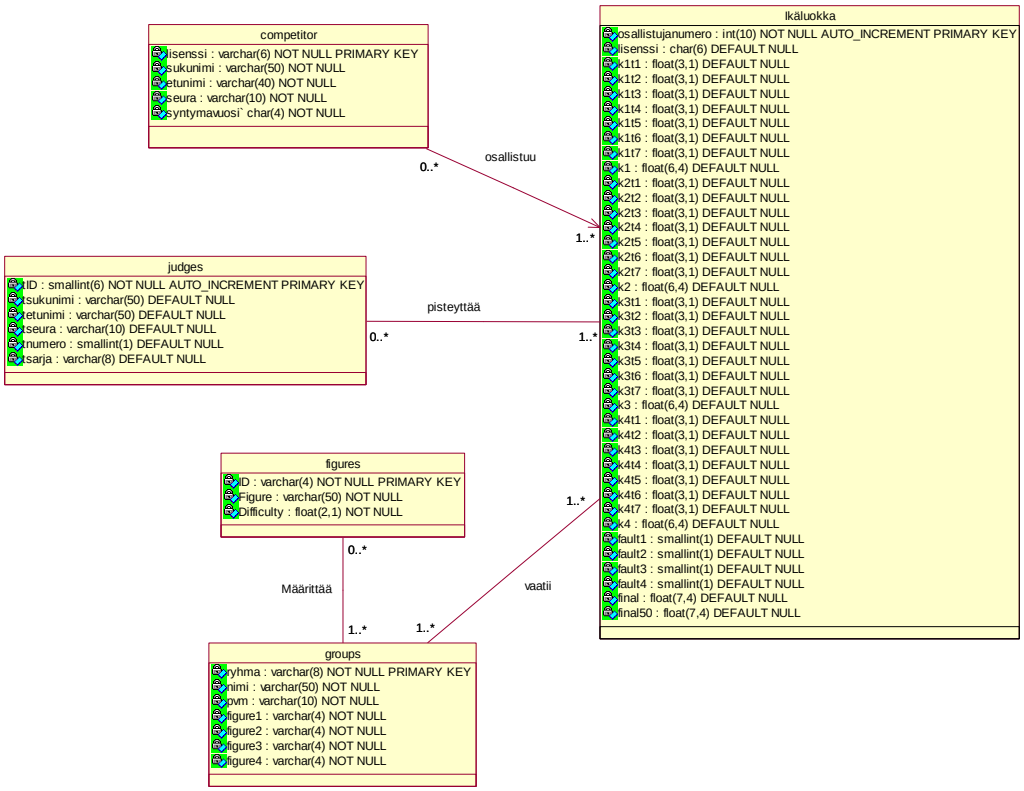
Ohjelmointikielen valinta käytiin Javan ja PHP:n välillä, koska ne ovat vahvasti esillä opintoohjelmassamme. Mielestämme PHP tarjosi järjestelmän kehittämiselle lähtökohtaisesti paremmat edellytykset kuin Javan kohdalla, koska molempien osaaminen oli PHP:n ohjelmointikielen osalta paremmalla tasolla Javaan verrattuna. Lisäksi oli ennakoitavissa, että ohjelmointitaidon oli kehityttävä edelleen joidenkin ratkaisuiden toteuttamiseksi; PHP:n ongelmatilanteisiin on mielestämme helpompi löytää ratkaisuvaihtoehtoja esimerkiksi foorumeilta tai kirjallisuudesta.

SQL-pohjainen tietokanta oli selkeä valinta tietokannaksi, koska koulutukseemme ei sisälly tutustumista muihin vaihtoehtoihin. Tietokannaksi valitsimme ilmaisen MySQL:n. Ainoaksi ongelmaksi jäi päättää, miten tietokantaa tultaisi hallinnoimaan. Vaihtoehdoista varteenotettavin olisi ollut Navicat tai SQLYog, mutta näiden kaupallisuuden takia päädyimme valitsemaan XAMPP:n. Se on ilmainen ja sitä on helppo hallinnoida PHPMyAdmin–käyttöliittymän avulla.

7.5 Tietokannan rakenne

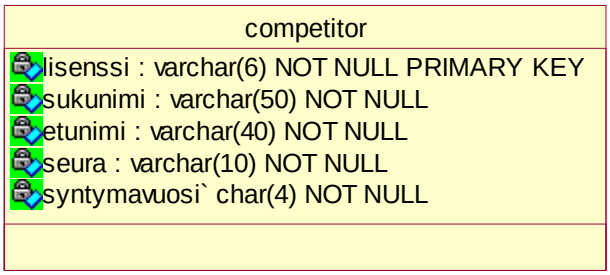
Pistelaskujärjestelmän rakenne koostuu järjestelmässämme seitsemästä taulusta yhden tietokannan sisällä. Jaoimme taulut selkeisiin, yksittäisiin kokonaisuuksiin, jolloin jokaisella taululla on oma tehtävänsä ja myöhemmissä kehitysvaiheissa on mahdollisimman helppoa jäsennellä käytettäviä tietoja. Kyseiset seitsemän taulua ovat nimeltään: competitor, figures, groups, judges, AG12, AG15 ja finajr.

Jokaiseen tauluun on luotu yksi tietokenttä, joka toimii pääavaimena (primary key). Tällä halusimme varmistaa mahdollisimman nopean toiminnallisuuden ilman erillisiä indeksointeja. Tietokannan rakenteesta (kuva 10) ilmenee taulujen väliset yhteydet ja riippuvuudet. Selkeyden vuoksi AG12, AG15 ja finajr -taulut ovat kuvassa niputettu yhdeksi Ikäluokka –nimiseksi tauluksi. Niputus on mahdollista, koska AG12, AG15 ja finajr -taulujen yhteydet muihin tauluihin ovat identtisiä eikä tauluilla ole keskenään yhteyttä.



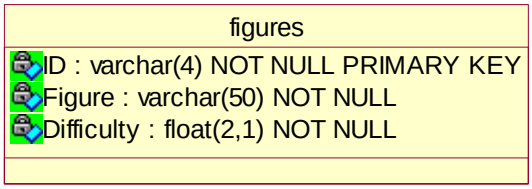
Kuva 10: Tietokannan rakenne.

Competitor-taulu sisältää seuraavat tiedot kilpailijoista: uimaliiton kilpailulisenssin lisenssinumeron, sukunimen, etunimen, edustettavan uimaseuran sekä uimarin syntymävuoden. Tämän taulun tiedot ovat pysyviä eli niitä ei nollata kilpailuiden välillä, ellei käyttäjä erikseen tätä toimenpidettä tee.



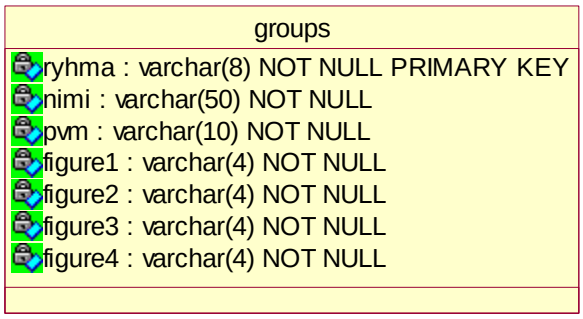
Kuva 11: Competitor-taulun rakenne.

Figures-taulussa määritetään kaikki mahdolliset uitavat taitouintikuviot kuviokilpailua varten. Figures-taulun sisältöä ei voida hallinnoida käyttöliittymän kautta, vaan kaikki mahdolliset muutokset tulee tehdä esimerkiksi PHPMyAdmin – tietokannan hallinnointityökalun avulla. Figures-taulussa on kaikkien kuvioiden tunnukset, niitä vastaavat nimet ja kunkin kuvion vaikeuskertoimet. Ohjelman luontihetkellä kuvioita oli 203 kappaletta.



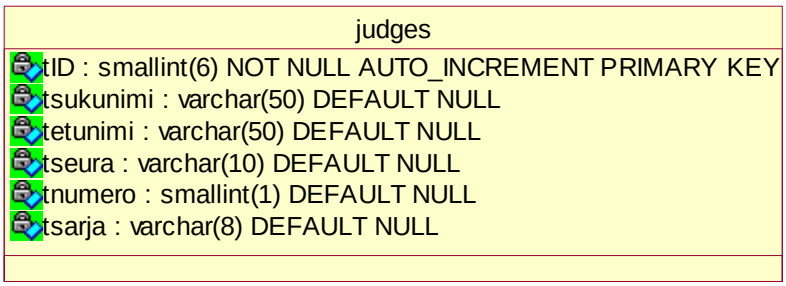
Kuva 12: Figures-taulun rakenne.

Groups-taulussa käsitellään kilpailtavan sarjan tietoja, joita ovat kilpailun nimi, päivämäärä ja kuviokilpailussa kilpailtavat kuviot. Kullekin riville määritellään kilpailtava ikäluokka (AG12, AG15 tai Fina jr.). Taulussa voi olla ainoastaan kolme riviä, joista kullekin ikäluokalle on varattu vain yksi rivi.



Kuva 13: Groups-taulun rakenne.

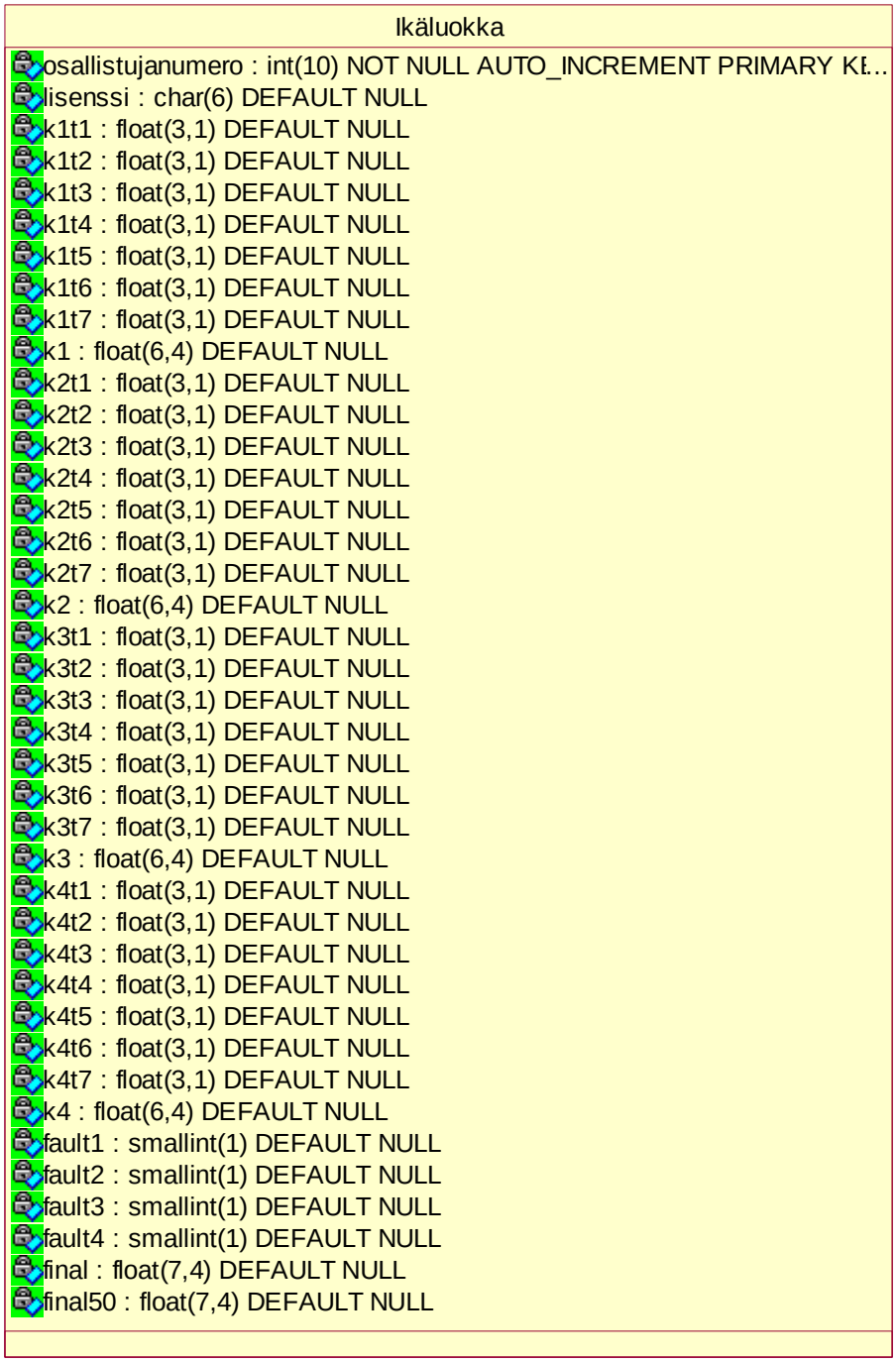
Judges-tauluun (Kuva 14.) tallennetaan kaikkien tuomareiden perustiedot, joita ovat sukunimi, etunimi, seura, tuomarinumero ja pisteytettävä sarja. Tämän lisäksi tauluun tallennetaan jokaiselle tuomarille tunnistenumero, minkä takia kyseisen tuomarin tietoja on helppoa käsitellä ohjelmassa. Judges-taulun toiminta on samankaltaista kuin groups-taulussa eli taulu tyhjennetään aina kilpiluiden välillä.



Kuva 14: Judges-taulun rakenne.

Jokaiselle kilpailtavalle ikäluokalle (AG12, AG15 ja Fina jr.) on olemassa rakenteeltaan identtiset taulut. Ikäluokkataulussa (Kuva 15.) käsitellään ainoastaan kilpailun pisteytykseen tarvittavia tietoja, joita ovat tuomarien antamat pisteet, näistä laskettavat kuvioiden yksittäiset

pisteet ja kuviokilpailun lopulliset pisteet. Tietokannassa ei tehdä laskutoimituksia vaan sinne ainoastaan lisätään palvelimella suoritettujen laskutoimitusten lopputuloksia. Pisteiden lisäksi Ikäluokkatauluun tallentuu sekä osallistujanumero, joka määräytyy kullekin uimarille ilmoittautumisjärjestyksen mukaan, että lisenssinumero, jolla sidotaan ilmoitettu uimari competitor-taulun kilpailijatietoihin.

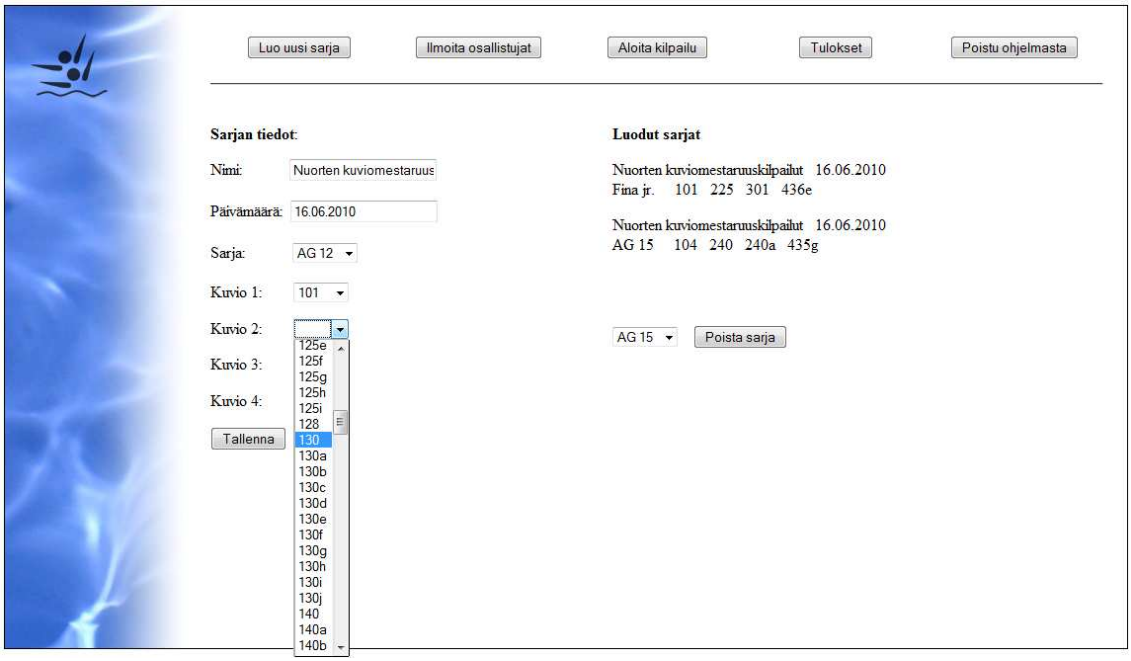


Kuva 15: AG12, AG15 ja Finajr -taulujen rakenne.

7.6 Järjestelmän ulkoasu

Järjestelmän ulkoasu pyrittiin pitämään mahdollisimman selkeänä ja yksinkertaisena. Ohjelman toimii valkoisella pohjalla, minkä ansiosta järjestelmän tekstit ovat selkeästi luettavissa. Elävöittääksemme järjestelmän mustavalkoista kokonaisuutta toimme väriä ja piristystä ulkoasuun lisäämällä vettä kuvaavan elementin käyttöliittymän vasempaan reunaan. Tämä sekä pehmentää käyttöliittymää että yhdistää järjestelmän uinnin maailmaan ajatustasolla.

Järjestelmä koostuu kolmesta pääkomponentista. Vasemman reunan tarkoitus on sekä navigoida etusivulle että elävöittää ja rajata järjestelmää. Yläpalkissa sijaitsee viisi painiketta, joiden avulla navigoidaan ohjelman eri toiminnallisuuksien välillä. Yläpalkin alapuolella on vaakaviivalla erotettuna ohjelman toimintoalue, jossa hallinnoidaan järjestelmän ydintoiminnallisuuksia.



Kuva 16: Järjestelmän ulkoasu – Sarjan perustaminen -toiminto.

7.7 Aikataulu

Aikatauluja tälle opinnäytetyölle oli kaksi erilaista. Ensimmäinen aikataulu on laadittu asiakkaan näkökulmasta eli käytännössä tässä määritettiin ohjelmistokehitysprojektin aikataulu. Toinen aikataulu käsitti opinnäytetyön tutkimuksen ja kirjallisen osuuden.

Asiakkaan eli Suomen uimaliiton toivomuksen mukaisesti sovimme järjestelmän kuvio-osuuden ensimmäisen käyttöversion toimituksen takarajaksi tammikuun ensimmäisen neljänneksen, että järjestelmän toiminnasta olisi ennen kauden alkua riittävästi testitilanteita ja käyttökokemuksia toimintavarmuuden varmistamiseksi. Ensimmäisen testiversio oli käytössä lokakuussa 2009 järjestetyissä taitouintikilpailuissa varsinaisen pistelaskun rinnalla. Järjestelmää kehitetään edelleen musiikkiohjelmien pisteytyksen osalta opinnäytetyön kirjallisen osuuden valmistumisen jälkeen. Opinnäytetyön tutkimuksellisen osuuden aikataulu eli opinnäytetyökurssin suunnitelmasta poiketen sekä kehitystyön että työelämän kiireiden ehdoilla. Tarkkaa aikataulua emme edellä mainituista tekijöistä johtuen määrittäneet. Viimeiseksi ja ehdottomaksi takarajaksi asetimme heinäkuun 2011 opinto-oikeudellisista syistä.

7.8 Riskit

Riskit jaoimme kahteen osaan: aikatauluriskeihin ja osaamiseen liittyviin riskitekijöihin. Aikataulun riskit keskittyivät lähinnä ohjelmistokehitysprojektin aikataulussa pysymiseen, johon vaikutti osaltaan myös osaamisen kehittyminen ja ongelmatilanteista selviäminen. Työkiireet ja sairastumiset olisivat voineet vaikuttaa negatiivisesti aikataulussa pysymiseen, minkä takia tähän varauduttiin järjestämällä aikaa kehitystyölle säännöllisesti ja riittävästi jo aikaisessa vaiheessa. Mahdolliset tekniset ongelmat kuten tietokoneen hajoaminen pyrittiin minimoimaan siten, että kaikki tiedot oli varmuuskopioitu vähintään kahteen ulkoiseen mediaan. Kehitysalustat oli myös asennettu usealle eri tietokoneelle. Huonoimmassakin mahdollisessa tilanteessa, olisimme voineet työstää järjestelmää koulun tarjoamilla välineillä.

Aikataulumme suurin riskitekijä oli ehdottomasti uuden tekniikan ja uusien menetelmien sisäistäminen. Uutta opittavaa oli paljon, joten hankalimpiin ongelmatilanteisiin ratkaisun löytämättä jättäminen olisi voinut pahimmassa tapauksessa kaataa koko projektin tai ainakin sotkea aikataulua pahasti.

8 Yhteenveto ja johtopäätökset

Opinnäytetyön tarkoitus oli tuottaa taitouinnin pistelaskujärjestelmä Suomen uimaliiton taitouintijaoston käytettäväksi ja samalla tutkia vaihejako- eli elinkaarimallien soveltuvuutta tällaisen ohjelmistoprojektin läpiviemiseen. Idea ohjelmiston kehittämisestä nousi esiin tuttavapiirin kontaktien kautta vuoden 2009 syksyllä. Asiakas esitti toiveen, jossa järjestelmän ensimmäinen versio saataisiin testattavaksi jo vuoden 2010 tammikuussa. Toiveen pohjalla oli ajatus saada toimiva järjestelmä kevään ikäkausimestaruuskilpailuihin. Kiireellisestä aikataulusta johtuen kehitimme järjestelmää samanaikaisesti tutkimustyön ohella, minkä takia kehitysvaiheessa ei voitu täysin hyödyntää tutkimuksen tuloksia.

Uuden järjestelmän kehittämismahdollisuus oli erittäin mielenkiintoinen haaste. Mielenkiintomme kehitysprojektia kohtaan lisääntyi entisestään saatuamme lähes vapaat kädet pistelaskujärjestelmän kehittämiseen. Tutkimustyön mielekkyys oli myös huomattavasti suurempi kuin aikaisemmissa kouluprojekteissa, koska tässä tapauksessa pystyimme hyödyntämään havaintojamme konkreettisella tasolla. Lisäksi ohjelmiston tarkoitusperät olivat aidosta toimintaympäristöstä tuottaen sille aitoa hyötyä, ja kohdeorganisaation toiminta oli valmiiksi lähellä siviilielämän intressejä. Mielestämme toimintakeskeinen opinnäytetyö palveli paremmin työelämään siirtymistä ja ammatillisen osaamisen kehittymistä kuin pelkkä tutkimuksellinen työ olisi kohdallamme palvellut. Koimme tämän tutkimuksen myötä saavamme hyvät valmiudet asiantuntijatehtäviin siirtymiseen. Teoreettisen tutkimustyön hyödyt ovat tärkeässä roolissa sekä opinnäytetyössä että työelämän prosessien ymmärtämisessä ja käytännön osaamisen hyödyntämisessä.

Henkilökohtainen osaamisemme kasvoi läpi koko opinnäytetyöprosessin. Lähtökohta ohjelmiston kehittämiselle teknisellä tasolla oli käytännössä täysin koulun oppien varassa, mikä edellytti jatkuvaa itsensä kehittämistä kirjallisuuden ja Internetin keskustelupalstojen avulla. Tiukan aikataulun takia oli helppo valita käytettävät työvälineet koulussa tutuiksi tulleista ohjelmointikielistä ja tietokannoista. Aloittaessamme työn tekemistä emme osanneet hahmottaa kokonaisvaltaisesti tarvittavaa työn määrää, ja tämä oli yllättäen todella suuri. Tämä opetti arvostamaan entisestä enemmän järjestelmällistä dokumentointia ja suunnitelmallisuutta ohjelmistokehitysprosesseissa. Onneksemme päätimme jo alkuvaiheessa toteuttaa työn omilla välineillämme, jolloin saatoimme työstää ketterästi työtä myös iltaisin ja viikonloppuisin koulun ollessa kiinni. Itse järjestelmän ohjelmointiin saimme koululta tukea, mutta varsinaista apua ongelmatilanteiden ratkaisemiseksi emme kuitenkaan saaneet. Tämä pakotti meidät paneutumaan entistä syvemmälle ohjelmointikielen antamiin mahdollisuuksiin ja myös ongelmatilanteisiin ja ohjelmointikielen vajaavaisuuksiin. Ongelmatilanteiden ratkaiseminen ilman koulun apua projektin eri vaiheissa lisäsikin motivaatiota ja itsevarmuutta luoda omia ratkaisuja.

Tutkimustyön perusteella valitsimme ohjelmistokehitysprojektillemme mielestämme sopivimmaksi vaihejakomalliksi prototyyppimallin. Suurin syy valinnalle oli vaatimusmäärittelyn suurpiirteellisyys, koska prototyypin avulla oli mahdollista herätellä ajatuksia, tarkentaa määrittelyitä ja luoda uusia toiminnallisuuksia. Lisäksi prototyyppimalli ei pidä sisällään vesiputousmallille tyypillistä ongelmaa, jossa paluu aikaisempiin kehitysvaiheisiin on lähes mahdotonta. Meille tämä mahdollisuus oli erittäin tärkeä, koska meillä ei ollut projektin alkuvaiheessa riittävää tietotaitoa eikä riittävän tarkkaa työksiantoa pystyäksemme luomaan tarpeeksi tarkat suunnitteludokumentit.

Ensimmäisten suunnitelmiemme mukaan evoluutiomalli vaikutti liian järeältä mallilta tällaiselle ohjelmistolle, mutta suunnitelmien ja vaatimusten tarkentuessa huomasimme evoluutiomallin periaatteiden olevan hyvin lähellä toimintatapojamme. Merkittävimpänä erona tekemisemme ja evo-mallin vaiheiden välillä oli dokumentoinnin määrä. Mielestämme dokumentoinnin merkitys korostuu suurissa projekteissa, joissa toimii useita eri toimijoita ja toiminnallisuuksien määrä on suurempi kuin kehittämässämme järjestelmässä. Sivuutimme myös spiraalimallin käytön heti projektimme alkuvaiheessa, koska huomasimme hyvin nopeasti, ettei ohjelmoinnin riskitekijöitä voitu tunnistaa etukäteen. Koska spiraalimalli pohjautuu riskien tunnistamiseen ja riskianalyysin korostamiseen, tuntui jo lähtökohtaisesti mahdottomalta käyttää kyseistä mallia.

RUP-malli antaa mahdollisuuden kehittää suuriakin ohjelmistoja. Muokkaamalla ja rajaamalla mallia omien tarpeiden mukaisesti olisi ollut mahdollista luoda tilanteeseen sopiva ja tarpeita vastaava toimintamalli, mutta tämäkin malli olisi vaatinut laajan kokonaiskuvan työkentästä sekä osaamista ja ymmärrystä itse ohjelmoinnista. Tämän takia emme valinneet RUP-mallia projektissamme käytettäväksi elinkaarimalliksi. Loppujen lopuksi erilaiset toimintamallit ovat kuitenkin hyvin lähellä toisiaan pääperiaatteiltaan. Erot malleissa syntyvät siis lähinnä eri asioiden painotuksista. Pienemmässä projektissa on helpompaa ottaa vapauksia ja poiketa kaavasta parhaan mahdollisen lopputuloksen saavuttamiseksi niin taloudellisesti kuin tuotteellisestikin. Olemmekin sitä mieltä, että mitä suurempi ohjelmistokehitysprojekti on kyseessä, sitä suuremmaksi dokumentaation ja järjestelmällisen mallin seuraamisen merkitys kasvaa.

Opinnäytetyön aikana törmäsimme useisiin erilaisiin haasteisiin, joista ehkä suurin yksittäinen osa-alue oli ohjelmistokehitysprosessin alkuvaiheessa osaamisemme taso. Koska ohjelmointitaitomme oli projektia aloittaessa huomattavasti heikommalla tasolla kuin loppuvaiheessa, oli haasteellista luoda tarkkoja suunnitelmia ja kuvauksia järjestelmästä. Tämä vaikutti järjestelmän ensimmäisten toiminnallisuuksien toteutukseen, sillä ohjelmointi piti suorittaa sen hetkisillä taidoilla pakottaen meidät yksinkertaisiin ja paikoin kankeisiin ratkaisuihin. Uuden oppiminen tapahtuikin pääosin kokeilemalla, minkä takia seuraavassa vaiheessa opittuja menetelmiä voitiin soveltaa aikaisemmin luotuihin toiminnallisuuksiin. Tällä pystyimme kehittä-

mään edelleen aikaisempia ratkaisuja, mutta tämä vaikutti negatiivisesti koodin luettavuuteen kommentoinnin puutteellisuuden ja ohjelmakoodin jäsentelyn kautta. Aikataulutus ja ajankäyttö loivat myös omalta osaltaan haasteellisia tilanteita.

Kuten aikaisemmin on jo tullut ilmi, oli ensimmäisen ohjelmistoversion aikataulu varsin tiukka. Tämä yhdessä henkilökohtaisten työ- ja koulukiireiden kanssa rajasi tutkimukselliseen työhön käytettävän ajan määrää, sillä suunnittelu- ja ohjelmointityö haukkasivat yllättävän suuren osan ajastamme. Jälkikäteen ajateltuna olisi varmasti ollut hyvä antaa tutkimukselliselle osuudelle enemmän aikaa, mikä olisi mahdollisesti voinut antaa pitkällä tähtäimellä nopeammin ja paremman lopputuloksen. Yllättävän suureksi ongelmaksi viimeisen järjestelmäversion valmistumisen jälkeen muodostui se, miten ohjelmisto saatettaisiin loppukäyttäjille mahdollisimman helposti. Ohjelmisto sisältää tietokantapalvelimen ja itse järjestelmän, jolloin ohjelmiston kooksi tuli kokonaisuudessaan yli 200 megatavua. Tämän vuoksi järjestelmän levittäminen ei onnistu sähköpostin välityksellä. Levittämisen lisäksi ongelmaa aiheutti asennusmedian luominen, koska käyttäjien tietokoneiden sisältö voi vaihdella suurestikin käyttöjärjestelmän ja muun ohjelmiston mukaan.

Yhteistyö asiakkaan kanssa lähti alusta asti toimimaan mielestämme erittäin hyvin. Saimme luotua hyvän luottamussuhteen asiakkaan kanssa, mikä ilmeni suunnittelun ja ohjelmoinnin osalta melko vapaana työskentelynä. Asiakas uskoi kykyymme tuottaa tarpeet täyttävä järjestelmä, mikä puolestaan loi meille halua ylittää asetetut odotukset. Kommunikointi ei tuottanut vaikeuksia missään vaiheessa, vaan saimme heti ensimmäisissä tapaamisissa selkeän tavoitteen ja suuntaviivat joiden puitteissa projektia tultaisiin työstämään. Lisäksi saimme selkeitä kehitystoiveita muun muassa oikeissa kilpailuissa toteutetuissa testaustilanteissa. Tällaisia testausmahdollisuuksia järjestettiin kehityksen eri vaiheissa. Toimiva yhteistyö tuottikin hedelmää, sillä saimme positiivista palautetta ohjelmiston toiminnallisuuksista ja helppokäyttöisyydestä. Järjestelmä on mahdollistanut aikaisempaa nopeamman kilpailun läpiviemisen ja samalla vähentänyt inhimillisten virheiden määrää lopullisten pisteiden laskennassa.

Yhteenvetona voidaan siis todeta, että ohjelmistokehitys on mielenkiintoinen ja lähes rajattomasti mahdollisuuksia antava ala. Nykypäivänä ei käytetä enää kaavamaisia vaihejakomalleja ohjelmistokehityksessä, vaan eri vaihejakomallien parhaita puolia voidaan soveltaa kuhunkin projektiin sopivimmilla tavoilla. Pistelaskujärjestelmän osalta voimme helposti todeta onnistuneemme tehtävässämme, koska asiakas on saanut toimintaansa selkeästi helpottavan apuvälineen, ja me olemme oppineet erittäin paljon.

Lähteet

Apache. 2009. Kotisivut. HTTP Server Project. Viitattu 6.1.2010. http://httpd.apache.org/

Ars technica. 2005. A history of the GUI. Viitattu 6.1.2010. http://arstechnica.com/old/content/2005/05/gui.ars/

DAU Press (Defense acquisition university press). 2001. Systems Engineering Funamentals. Viitattu 1.4.2010. http://www.dau.mil/pubscats/PubsCats/SEFGuide%2001-01.pdf

FINA (Kansainvälinen uimaliitto). 2009. Taitouinnin säännöt. Viitattu 2.12.2009. http://www.fina.org/project/index.php?option=com_content&task=view&id=49&Itemid=119

Gilmore, W. J. 2005. PHP & MySQL – Tehokas hallinta. Jyväskylä: Gummerus Kirjapaino Oy.

Haikala, I. & Märijärvi, J. 2004. Ohjelmistotuotanto. 10. painos. Helsinki: Talentum.

Halonen, R. Tietojärjestelmän vaihtaminen, Tapaustutkimus. Viitattu 19.5.2010. http://www.tkts.fi/lehti/a20/halonen.pdf

Heinisuo, R. & Rauta, I. 2007. PHP ja MySQL – Tietokantapohjaiset verkkopalvelut. 4. Painos. Helsinki: Talentum.

Julkisen hallinnon suositus (JHS-suositukset). 2010. Viitattu 1.4.2010. http://docs.jhs-suositukset.fi/jhs-suositukset/JHS165/JHS165.html

Koskimies, K. & Mikkonen, T. 2005. Ohjelmistoarkkitehtuurit. Jyväskylä: Gummerus Kirjapaino Oy.

Laine, H. 2002. Johdatus sovellussuunnitteluun. Viitattu 16.11.2009. http://www.cs.helsinki.fi/u/laine/jossu/s02/jss_u1.pdf

Lemay, L. 1998. WWW-julkaiseminen HTML 4. Jyväskylä: Gummerus kirjapaino Oy.

May, E. L & Zimmer, B. A. 1996. The evolutionary development model for software Viitattu 24.11.2009. http://www.hpl.hp.com/hpjournal/96aug/aug96a4.pdf

MikroPC. 2004. Viitattu 10.5.2010. http://mikropc.net/nettilehti/pdf/1706200452.pdf

MySQL. 2010. Kotisivut. Viitattu 11.5.2010. http://www.mysql.com/

NCSA (The National Center for Supercomputing Applications). Viitattu 7.12.2009. http://hoohoo.ncsa.illinois.edu/cgi/intro.html/

Ohjelmistoprosessit ja ohjelmistojen laatu. Viitattu 16.11.2009. http://www.cs.helsinki.fi/u/taina/opol/k-2009/pdf/luku-6_2.pdf

Oulun seudun ammattikorkeakoulu. (OAMK) 2010. Viitattu 8.9.2010. http://www.students.oamk.fi/~s6matu00/sekalaista/ot/teema2.ppt

Oulun seudun ammattikorkeakoulu (OAMK). 2009. Viitattu 23.11.2009. http://www.oamk.fi/sbc/ohjelmistotuote/ohjelmistotuotanto/vesiputous.gif

PHP. 2009. Kotisivut. Viitattu 7.12.2009 http://www.php.net/usage.php

Pohjonen, R. 2002. Tietojärjestelmien kehittäminen. Jyväskylä: Docendo.

Polvinen, T. 1999. Tietokannat käytännön työssä. Porvoo: WSOY.

Qastation. 2009. Viitattu 23.11.2009.
http://qastation.wordpress.com/2008/04/26/software-development-life-cycles-part-4/

Rational Unified Process. Viitattu 8.9.2010.
http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf
Suomen Uimaliitto. 2009 a. Viitattu 11.10.2009 http://www.uimaliitto.fi/fi/uimaliitto/

Suomen Uimaliitto. 2009 b. Taitouinti. Viitattu 2.12.2009.
http://www.uimaliitto.fi/fi/taitouinti/esittely/index.php/

XAMPP. 2010. Kotisivut Viitattu 11.4.2010. http://www.apachefriends.org/en/xampp.html

Kuvat ja kuviot

Kuva 1: PHP:n käytön kehitys. 11
Kuva 2: Vesiputousmalli..... 14
Kuva 3: Prototyyppimalli 15
Kuva 4: Spiraalimalli..... 16
Kuva 5: EVO-malli – Perinteinen vesiputousmalli (a) verrattuna Evo-malliin (b) 17
Kuva 6: RUP –mallin vaiheiden limittyminen 18
Kuva 7: Esitutkimuksen (esiselvitys) kulku 20
Kuva 8: Vaatimusmäärittelyn eri vaiheet 22
Kuva 9: Kysymyslomakkeen vastausten indeksiarvot. 33
Kuva 10: Tietokannan rakenne. 35
Kuva 11: Competitor–taulun rakenne. 35
Kuva 12: Figures–taulun rakenne. 36
Kuva 13: Groups–taulun rakenne. 36
Kuva 14: Judges–taulun rakenne. 36
Kuva 15: AG12, AG15 ja Finajr –taulujen rakenne. 37
Kuva 16: Järjestelmän ulkoasu – Sarjan perustaminen -toiminto. 38

Liitteet

Liite 1: Vaatimusmäärittelyraportti 47
Liite 2: Kysymyslomake 57
Liite 3: Käyttötapauskuvaukset..... 58
Liite 4: Kaaviot: 64
Liite 5: Esimerkki ohjelmistokoodista 67

Liite 1: Vaatimusmäärittelyraportti

Pistelaskujärjestelmä

Versio 1.0

Sisällys

1 Johdanto 49
2 Yleiskuvaus 50
3 Tiedot ja tietokanta 50
3.1 Tietokanta 50
3.1.1 Taulut ag12, ag15 & finajr 51
3.1.2 Competitor 52
3.1.3 Figures 52
3.1.4 Judges 52
3.1.5 Groups 53
3.2 Tiedostot ja asetustiedostot 53
4 Toiminnot 53
4.1 Yleiset toiminnallisuudet 53
4.2 Uuden sarjan perustaminen 53
4.3 Ilmoittaminen 53
4.3.1 Osallistujien ilmoittaminen 54
4.3.2 Tuomareiden ilmoittaminen 54
4.4 Kilpailun aloittaminen 54
4.5 Tulosteet 55
4.6 Järjestelmän sulkeminen 55
5 Muut ominaisuudet 55
5.1 Suorituskyky ja vasteajat 55
5.2 Ylläpidettävyys 55
5.3 Siirrettävyys ja yhteensopivuus 55
6 Suunnittelurajoitteet 55
6.1 Tietoturvallisuus 55
6.2 Laitteistorajoitteet 56
6.3 Ohjelmistorajoitteet 56
7 Jatkokehitysajatuksia 56

1 Johdanto

Dokumentin tarkoituksena on tuottaa kattava suunnitelma siitä, mitä ominaisuuksia rakennettavassa järjestelmässä tulisi olla. Tässä vaiheessa asetetaan raamit, joiden mukaan ohjelmistoa lähdetään kehittämään.

Ohjelmiston käyttötarkoituksena on helpottaa toimitsijoiden työtä taitouintikilpailuissa tarjoamalla käytännöllisen tavan merkitä ja tallentaa pisteytyksiä. Tarkoituksena on lisäksi tuottaa työkalu, jonka avulla voidaan vähentää manuaalisen työn määrää ja yhdenmukaistaa Suomen Uimaliiton alaisuudessa pidettävien kilpailujen tulosteita. Kehitettävää ohjelmistoa tullaan siis käyttämään uimahalleissa ympäri Suomen.

Tässä vaatimusmäärittelyssä käydään läpi järjestelmän yleiskuvaus, tietokantarakenne, järjestelmän toiminnallisuudet ja vaatimukset. Viimeisessä kappaleessa käydään läpi vielä mahdollisia jatkokehityssuuntia.

2 Yleiskuvaus

Järjestelmän toiminta ei saa olla riippuvainen Internet-yhteydestä. Se tullaan kehittämään verkosta riippumattomaksi järjestelmäksi, koska ohjelmaa tullaa käyttämään pääasiallisesti uimahalleissa, joissa ei välttämättä ole mahdollisuutta. liittää tietokonetta verkkoon. Järjestelmän ulkoasu pyritään pitämään pelkistettynä, jolloin vähäisemmänkin kokemuksen omaava käyttäjä pystyy käyttämään järjestelmää mahdollisimman sujuvasti ja ongelmitta. Käyttäjien taustat vaihtelevat toisistaan paljon, joten tietoteknisen osaamisen erot täytyy myös ottaa huomioon.

Tietojen tallentaminen tietokantaan tulee tapahtua riittävän nopealla syklillä, jolloin mahdollisien virtakatkoksien aiheuttamat tietojen menetykset voidaan minimoida. Yleisesti ottaen tilanne on kuitenkin turvattu, sillä useimmissa tapauksissa järjestelmää tullaan käyttämään kannettavilla tietokoneilla, joissa akku huolehtii virran saannista virtakatkoksien aikana.

Järjestelmä kehitetään käyttämään selainta, jolloin ulkoasu vaihtelee käytettävän selaimen asetuksista riippuen. Pääasiallisena testausselaimena käytetään Mozilla Firefoxia, koska kyseinen selain toimii sekä Windows- että Linux – käyttöjärjestelmässä.

Järjestelmä ei siis tule vaatimaan ulkoisia eikä laitteistoliittymiä toimiakseen. Ainoa tarvittava liittymä on ohjelmistoliittymät: Internet-selain ja Xampp – kehitysympäristö, joka tarjoaa palvelimen ja tietokannan.

3 Tiedot ja tietokanta

Tässä kappaleessa käydään läpi suunniteltavan järjestelmän tietokantarakenne. Tietokantoihin rakennetaan erilaisia tarkoituksia varten useita eri tauluja, joihin voidaan lisätä tarkoituksesta riippuen erilaista dataa myöhäisempää käsittelyä varten.

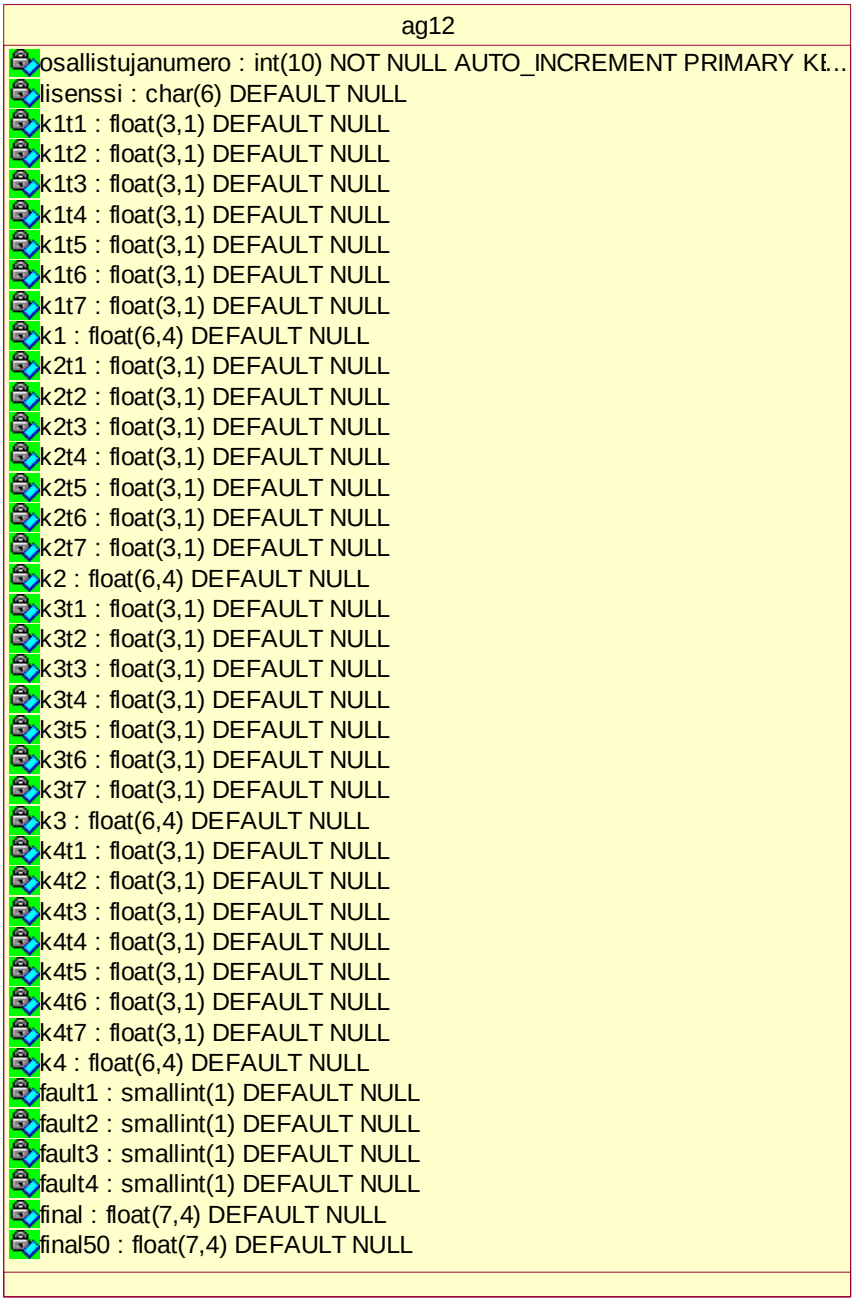
3.1 Tietokanta

Tietokanta sisältää seitsemän eri taulua, joiden sisältö on selvitetty tarkemmin myöhemmin. Taulut tyhjennetään aina uutta kilpailua aloitettaessa pl. competitor ja figures, jotka sisältävät pysyvää tietoa. Tietokanta rakennetaan MySQL – ohjelmistoa käyttäen.

3.1.1 Taulut ag12, ag15 & finajr

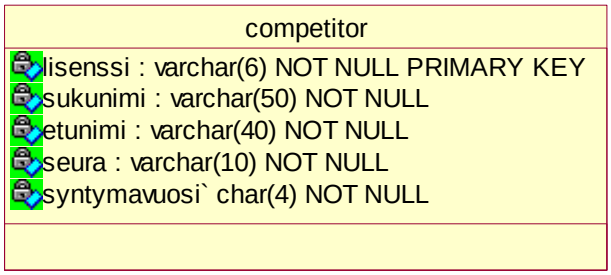
Nämä kolme taulua ovat identtisiä ja ne pitävät sisällään kunkin kilpailtavan ikäluokan kilpailuun liittyvät tiedot. Taulun perusavaimena toimii ilmoitetun uimarin juokseva osallistujanumero. Näin jokaisella uimarilla on henkilökohtainen osallistujanumero, johon voidaan sitoa kilpailusta seuraavat muut tiedot.

Muita taulun sarakkeita ovat, lisenssi, tuomarikohtaiset pisteet kuvioittain, kuvioiden pisteet, kuviokohtaiset virhepisteet, lopulliset pisteet sekä musiikkiohjelmia varten tallentuvat kuviopisteet. Lisenssi toimii taulussa viiteavaimena, joka sitoo osallistujanumeron tiettyyn uimariin.



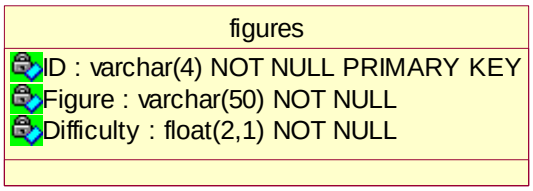
3.1.2 Competitor

Competitor -tauluun viedään kaikkien uimareiden tiedot. Sen perusavaimena käytetään saraketta lisenssi. Jokaisella uimarilla on uniikki Suomen Uimaliiton määrittelemä lisenssinumero, joten tietoa voidaan käyttää yhtä ainutkertaisena kuin esimerkiksi sosiaaliturvatunnusta. Muita tauluun luotavia sarakkeita ovat: sukunimi, etunimi, syntymävuosi sekä seura. Seura -sarakkeeseen tallennetaan uimarin uimaseuran nimen lyhenne.



3.1.3 Figures

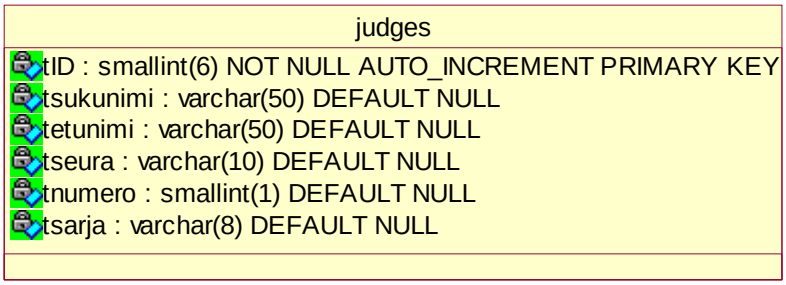
Tämä taulu pitää sisällään tiedot kaikista taitouinnissa käytössä olevista kuvioista. Perusavaimena toimii sarake ID, joka on kuvion kansainvälisen uimaliiton määrittelemä tunnistenumero. Lisäksi taulussa on sarakkeet kuvion nimelle sekä vaikeuskertoimelle.



3.1.4 Judges

Tuomareiden taulun tärkein tehtävä on luoda mahdollisuus tarkastella kilpailun jälkeen yksittäisen tuomarin pisteytystä. Taulun perusavain on tID, jonka numerointi asetetaan juoksevaksi.

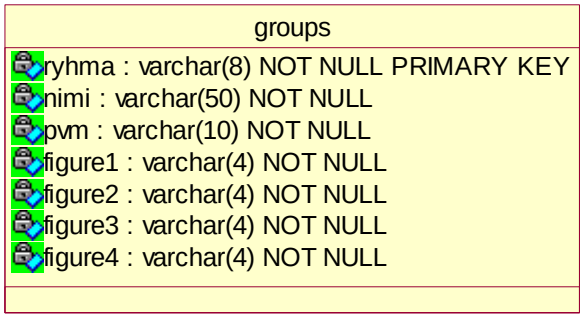
Muita sarakkeita ovat tsukunimi, tetunimi, tseura, tnumero sekä tsarja. Näillä tiedoilla voidaan kertoa tuomarista oleellisimmat tiedot. Tnumero määräytyy kilpailuun ilmoitettujen tuomarien määrän ja tuomarointipaikan sijainnin mukaan. Tsarja -sarakkeeseen ilmoitetaan arvo ag12, ag15 tai finajr, jotta tiedetään mitä ikäluokkaa kukin tuomari tuomitsee.



3.1.5 Groups

Groups –tauluun lisätään tiedot perustettavista kilpailuista. Kullekin ikäluokalle perustetaan oma kilpailu, joten pääavaimeksi valikoituu sarake ryhma. Ryhma sarakkeeseen asetetaan jokin kolmesta (ag12, ag15 & finajr) ikäluokasta.

Kilpailun muita oleellisia tietoja ovat nimi, pvm (päivämäärä) ja neljä ennalta määrättyä uitavaa kuviota (figure1, figure2, figure3 & figure4).



3.2 Tiedostot ja asetustiedostot

Järjestelmä koostuu useista eri tyyppisistä tiedostoista. Suurin osa tiedostoista tulee Xampp –kehitysympäristöpaketin mukana. Tämän lisäksi järjestelmään tullaan luomaan tarvittava määrä käyttöliittymän vaatimia .php –tiedostoja sekä tietokantatiedostoja.

4 Toiminnot

Tässä kappaleessa käydään läpi kaikki järjestelmän tärkeimmät toiminnallisuudet yksityiskohtaisesti sivunäkymä kerrallaan.

4.1 Yleiset toiminnallisuudet

Järjestelmän yläreunaan tullaan sijoittamaan navigointipalkki, jonka avulla voidaan siirtyä luomaan uusi sarja, ilmoittamaan osallistujia, aloittaa kilpailu, tarkastella tulosteita ja sulkea järjestelmä. Lisäksi vasempaan yläkulmaan sijoitetaan ohjelman logo, jota painamalla pääsee suoraan järjestelmän etusivulle.

4.2 Uuden sarjan perustaminen

Perustettaessa uutta sarjaa järjestelmä kysyy sarjan oleellisimmat tiedot, jotka ovat nimi, päivämäärä, ikäluokka ja kilpailtavat kuviot. Ikäluokka sekä kilpailtavat kuviot valitaan tietokannasta tulostettavasta alasvetovalikosta. Kilpailun tiedot viedään tietokantaan painamalla tallenna –nappia. Tietojen viemisen jälkeen sivulle ilmestyy perustetun sarjan tiedot sekä mahdollisuus poistaa aikaisemmin luotu sarja valitsemalla poistettava sarja alasvetovalikosta ja painamalla poisto -nappia. Järjestelmä vaatii käyttäjältä poistettaessa vahvistusta toiminnalle.

Järjestelmä ei anna lisätä tietoja tietokantaan, jos ikäluokan tunnusta ei ole määritetty tai kyseinen ikäluokka on jo lisätty tietokantaan.

4.3 Ilmoittaminen

Ilmoittamisnäkymässä voidaan siirtyä hallinnoimaan luotujen sarjojen osallistujaluetteloita sekä tuomareita.
Osallistujaluetteloa pääsee muokkaamaan valitsemalla alasvetovalikosta aikaisemmin perustettu sarja ja painamalla siirtymisnappia. Jos yhtään sarjaa ei löydy tietokannasta, on alasvetovalikko tyhjä jolloin siirtymisnapin painallus ei tee mitään.

Tuomareiden ilmoittaminen tapahtuu samalla tavalla kuin osallistujienkin. Poikkeuksena on toinen alasvetovalikko, jossa määritetään sarjaan ilmoitettavien tuomareiden määrä. Tuomareiden määrä voi vaihdella kolmen ja seitsemän välillä.

4.3.1 Osallistujien ilmoittaminen

Tässä näkymässä voidaan viedä tietokantaan uusien uimareiden tiedot sekä viedä sarjaan osallistuvien uimareiden lisenssinumerot kyseisen sarjan tauluun tietokannassa. Lisäksi näkymässä voidaan poistaa ilmoitettu uimari sarjasta tai kaikki uimarin tiedot tietokannasta. Uutta uimaria lisättäessä järjestelmä vaatii seuraavat tiedot: Lisenssinumero, sukunimi, etunimi, uimarin edustama seura sekä syntymävuosi. Nämä tiedot syötetään tekstinä näille varattuihin tekstikenttiin. Tiedot viedään tietokantaan painamalla lisäysnappia. Lisenssinumero on jokaisella uimarilla henkilökohtainen, joten järjestelmä ei anna lisätä tietokantaan kahta identtistä lisenssinumeroa.

Uimarin kaikki tiedot voi poistaa tietokannasta syöttämällä poistamiskohdan tekstikenttään uimarin lisenssinumero ja painamalla poistonappia.Uimarin ilmoittaminen kilpailuun tapahtuu valitsemalla alasvetovalikosta aikaisemmin tietokantaan lisätty uimari. Alasvetovalikko näyttää uimarista nimen, seuran ja lisenssinumeron, jolloin voidaan olla varmoja, että kyseessä on oikea uimari. Lisääminen tapahtuu painamalla ilmoittamisnappia. Järjestelmä tulostaa ilmoituksen lisäyksen onnistumisesta ja lisää uimarin tiedot ilmoitettujen uimareiden listaan.

Ilmoittautumisen voi perua valitsemalla ilmoitettu uimari perumisen alasvetovalikosta, johon tulostuu kaikki ilmoitetut uimarit. Tämän jälkeen painetaan peruutusnappia ja järjestelmä poistaa uimarin kyseisestä sarjasta.

4.3.2 Tuomareiden ilmoittaminen

Näkymään tulostuu jokaiselle ilmoitettavalle tuomarille oma rivi, johon syötetään sukunimi, etunimi ja seura. Rivien lukumäärä määräytyy ilmoittamissivulla määritetyn tuomarien lukumäärän mukaan. Tiedot ilmoitetaan tekstikenttiin kirjoittamalla. Tietokantaan tiedot viedään painamalla lisäysnappia. Lisäämisen jälkeen järjestelmä siirtyy takaisin ilmoittamissivulle. Vaihtoehtoisesti voidaan siirtyä takaisin edelliselle sivulle painamalla palaa –nappia, jolloin tietokantaan ei lisätä uusia tuomareita.

Jos tuomareita yritetään lisätä sarjaan, johon on jo aikaisemmin lisätty tuomareita, tulostuu aikaisemmin ilmoitettujen tuomareiden tiedot. Tässä kohdassa järjestelmä antaa mahdollisuuden poistaa aikaisemmin ilmoitetut tuomarit tai jatkaa kyseisillä tuomareilla. Molemmille toiminnallisuuksille on omat nappinsa.

4.4 Kilpailun aloittaminen

Näytölle tulostuu alasvetovalikko perustetuista ikäluokista, joista valitaan aloitettava sarja. Painamalla aloittamisnappia järjestelmä siirtyy kyseisen sarjan ensimmäiseen kuvioon. Näytölle tulostuu alasvetovalikko, josta valitaan kuvion aloittava uimari. Tämän lisäksi näytölle tulostuu jokaiselle kilpailuun ilmoitetulle tuomarille oma tekstikenttä, johon kirjataan tuomarin antamat pisteet. Tämän lisäksi tulostuu vielä yksi tekstikenttä mahdollisille virhepisteille. Kaikki ilmoitetut tiedot lisätään tietokantaan painamalla lisäysnappia. Järjestelmä käy läpi kaikki sarjan uimarit samaa kaavaa noudattaen pl. aloittavan uimarin valinta, jonka jälkeen järjestelmä ilmoittaa kuvion pisteytyksen olevan valmis. Seuraavaan kuvioon siirrytään painamalla seuraava –nappia. Kun kaikki kuviot on pisteytetty, ilmoittaa järjestelmä kilpailun loppumisesta ja painamalla valmis –nappia järjestelmä laskee pisteet ja siirtyy kilpailun aloittamissivulle.

4.5 Tulosteet

Tulosteet sivulla on kaksi toiminnallisuutta: Siirtyminen tarkastelemaan osallistujaluetteloita sekä lopullisia tuloksia. Kullekin sarjalle on omat nappinsa niin osallistujaluettelolle kuin tuloksillekin. Painamalla jotakin näistä napeista avautuu selaimeen uusi ikkuna, joka pitää sisällään halutun listan. Tulosteista valittavana ovat ainoastaan valmiit listat, jolloin esimerkiksi keskeneräisen kilpailun tuloksia ei ole saatavilla.

4.6 Järjestelmän sulkeminen

Tässä näkymässä järjestelmä vaatii käyttäjältä vahvistuksen järjestelmän sulkemiseen. Painamalla ”Kyllä” järjestelmä sulkee tietokanta- sekä palvelinyhteyden. Valitsemalla ”Ei” järjestelmä siirtyy etusivulle.

5 Muut ominaisuudet

Kappaleen tarkoitus on määrittää projektin aikaisemman lisäksi mainitsemisen arvoisia muita ominaisuuksia. Kappaleessa käydään läpi järjestelmän suorituskykyyn ja vasteaikoihin, ylläpidettävyyteen sekä siirrettävyyteen ja yhteensopivuuteen vaikuttavia tekijöitä.

5.1 Suorituskyky ja vasteajat

Järjestelmä tulee käsittelemään suhteellisen pieniä tietomääriä kerralla, joten sen suorituskyky ja vasteajat pysyvät hyvällä tasolla, kunhan tietokone täyttää Xampp –kehitysympäristön asettamat laitteistovaatimukset.

5.2 Ylläpidettävyys

Järjestelmän toiminnallisuuksia ei tarvitse aktiivisesti ylläpitää. Tarvittavat lisäykset tietokantaan tehdään järjestelmän käyttöliittymän kautta, joten kuka tahansa pystyy itse hallitsemaan järjestelmän kaikkia tarvittavia tietoja. PHPmyadmin työkalun avulla on mahdollista tehdä tietokantarakenteeseen muutoksia ja php-koodia voi muokata millä tahansa tekstieditorilla, koska koodi on jätetty avoimeksi.

5.3 Siirrettävyys ja yhteensopivuus

Järjestelmästä kehitetään asennusmedia, jonka avulla pistelaskujärjestelmän voi asentaa mille tahansa Windows XP:tä uudemmalle Windows käyttöjärjestelmälle. Järjestelmä luodaan ensisijaisesti käytettäväksi Windows työympäristöön, mutta käytettävät työkalut ovat Unix-järjestelmien kanssa yhteensopivia. Järjestelmän ensimmäinen versio optimoidaan Mozilla Firefox –selaimelle ja 1024x768 resoluutiolle. Ohjelmisto tulee myös toimimaan muillakin selaimilla ja resoluutioilla vähintäänkin tärkeimpien toiminnallisuuksien osalta.

6 Suunnittelurajoitteet

Suunnittelurajoitteet – kappale pitää sisällään mahdolliset rajoitteet sekä ohjelmistopuolella että laitteistopuolella. Tämän lisäksi kappaleessa määritellään merkittävimmät tietoturvariskit.

6.1 Tietoturvallisuus

Järjestelmän kehittämisessä ei tarvitse huolehtia tietoturvallisuudesta järjestelmään ilmoitettavien tietojen osalta, koska kaikki käsiteltävä tieto on julkista.

6.2 Laitteistorajoitteet

Laitteistorajoitteet rajoittuvat pääosin Xampp –kehitysympäristön laitteistovaatimuksiin:

- 128 Mt keskusmuistia
- 300 Mt vapaata kovalevytilaa
- Windows 2000, XP, Vista, Windows 7
- 32 bittinen käyttöjärjestelmä

6.3 Ohjelmistorajoitteet

Asennusmedia tulee sisältämään kaiken muun tarvittavan ohjelmiston paitsi Internet-selaimen.

7 Jatkokehitysajatuksia

Järjestelmää tullaan kehittämään edelleen loppukäyttäjien kommenttien ja parannusehdotusten myötä. Järjestelmään tullaan myös lisäämään kuviopistelaskun lisäksi musiikkiohjelmien pisteytysosio. Toinen jatkokehitysmahdollisuus on järjestelmän testaaminen ja kehittäminen myös Unix-käyttöjärjestelmille yhteensopivaksi.

Liite 2: Kysymyslomake

Kuhmonen & Nikkanen 2009
Opinnäytetyön kysymyslomake
Case Taitouinnin pistelaskujärjestelmä / Suomen Uimaliitto / Kilpailusihteeri
1. Sukupuoli: [ ] Mies [ ] Nainen
2. Ikä: [ ] alle 20 [ ] 20-29 [ ] 30-39 [ ] 40-49 [ ]50-59 [ ] 60-

Seuraavissa kysymyksissä rastita **yksi (1)** parhaiten sopiva vaihtoehto.

3. Kuinka paljon tarvetta olisi taitouinnin pistelaskujärjestelmän kehittämiselle?
[ ] Paljon [ ] Melko paljon [ ] Vähän [ ] Ei lainkaan
4. Oletteko käyttänyt aikaisemmin jotakin taitouinnin pistelaskujärjestelmää?
[ ] Paljon [ ] Melko paljon [ ] Vähän [ ] Ei lainkaan
5. Kuinka paljon ohjelman helppokäyttöisyys merkitsee?
[ ] Paljon [ ] Melko paljon [ ] Vähän [ ] Ei lainkaan
6. Kuinka tärkeää on, että järjestelmä laskee uimarien pisteet itsenäisesti?
[ ] Paljon [ ] Melko paljon [ ] Vähän [ ] Ei lainkaan
7. Kuinka tarpeellista on, että uimarien tiedot tallentuvat järjestelmään?
[ ] Paljon [ ] Melko paljon [ ] Vähän [ ] Ei lainkaan
8. Kuinka tarpeellista on , että tuomareiden tiedot tallentuvat järjestelmään?
[ ] Paljon [ ] Melko paljon [ ] Vähän [ ] Ei lainkaan
9. Kuinka tärkeää, että ohjelma on suomenkielinen?
[ ] Paljon [ ] Melko paljon [ ] Vähän [ ] Ei lainkaan
10. Kuinka tärkeää on ohjelman visuaalinen ulkoasu?
[ ] Paljon [ ] Melko paljon [ ] Vähän [ ] Ei lainkaan
11. Kuinka tarpeellista on, että ohjelmaa voi käyttää ilman Internet yhteyttä?
[ ] Paljon [ ] Melko paljon [ ] Vähän [ ] Ei lainkaan
12. Kuinka tärkeää on, että ohjelmasta voi tulostaa lopulliset tulokset?
[ ] Paljon [ ] Melko paljon [ ] Vähän [ ] Ei lainkaan
13. Kuinka tärkeää on, että tuloksissa on eriteltyinä kuviokohtaiset tulokset?
[ ] Paljon [ ] Melko paljon [ ] Vähän [ ] Ei lainkaan
14. Kuinka tärkeää on, että tuloksissa on eriteltyinä tuomarikohtaiset tulokset?
[ ] Paljon [ ] Melko paljon [ ] Vähän [ ] Ei lainkaan
15. Kuinka tärkeää on järjestelmän helppo asennus?
[ ] Paljon [ ] Melko paljon [ ] Vähän [ ] Ei lainkaan

Kiitos vaivannäöstänne!

Liite 3: Käyttötapauskuvaukset

Käyttötapauskuvaukset

Pistelaskujärjestelmä

Versio 1.0

SISÄLLYS

1 Sarjan perustaminen 60
2 Sarjan poistaminen 60
3 Kilpailijoiden ilmoittaminen 60
4 Uimarin lisääminen tietokantaan 60
5 Uimarin poistaminen tietokannasta..... 60
6 Uimarin ilmoittaminen kilpailuun 61
7 Ilmoittautumisen peruminen..... 61
8 Tuomareiden ilmoittaminen 61
9 Tuomarin ilmoittaminen sarjaan. 62
10 Kilpailun aloittaminen..... 62
11 Tulosteiden tarkastelu 63
12 Järjestelmän sulkeminen 63

1 Sarjan perustaminen

Yhteenveto: Kilpailusihteeri voi perustaa sarjan ja ilmoittaa sille tarvittavat perustiedot.

Kuvaus: Käyttäjä ilmoittaa perustettavan kilpailun nimen, päivämäärän kilpailevan ikäluokan sekä neljä kilpailussa kilpailtavaa kuviota.

Poikkeukset: Jos sarja on jo perustettu, antaa järjestelmä virheilmoituksen aiheesta.

2 Sarjan poistaminen

Yhteenveto: Käyttäjä poistaa aikaisemmin luodun sarjan.

Kuvaus: Käyttäjä valitsee alasvetovalikosta luotujen sarjojen listasta poistettavan sarjan sekä poistaa sarjan tiedot tietokannasta poistamisnapilla.

Poikkeukset: Jos sarjoja ei ole luotu, ei poistamisikkunaa ole näkyvissä.

3 Kilpailijoiden ilmoittaminen

Yhteenveto: Käyttäjä ilmoittaa uimareita kilpailtaviin sarjoihin.

Kuvaus: Käyttäjä valitsee alasvetovalikosta aiemmin luodun sarjan ja siirtyy kyseisen sarjan ilmoittamissivulle.

Poikkeukset: Jos sarjoja ei ole luotu, alasvetovalikossa ei ole valittavissa sarjoja.

4 Uimarin lisääminen tietokantaan

Yhteenveto: Käyttäjä lisää uimarin tietokantaan.

Kuvaus: Käyttäjä lisää uimarin lisenssinumeron, sukunimen, etunimen, seuran ja syntymävuoden tietokantaan ja tallentaa tiedot painamalla tallennusnappia.

Poikkeukset: Jos uimarin lisenssinumero löytyy jo tietokannasta, tiedot eivät tallennu tietokantaan.

5 Uimarin poistaminen tietokannasta

Yhteenveto: Käyttäjä poistaa aikaisemmin luodun uimarin tiedot tietokannasta.

Kuvaus: Käyttäjä kirjoittaa tekstikenttään poistettavan uimarin lisenssinumeron ja painaa poistamisnappia, jolloin järjestelmä poistaa kaikki kyseisen uimarin tiedot tietokannasta.

Poikkeukset: Ei

6 Uimarin ilmoittaminen kilpailuun

Yhteenveto: Käyttäjä ilmoittaa uimareita kilpailtavaan ikäluokkaan.

Kuvaus: Käyttäjä valitsee alasvetovalikosta tietokantaan syötetyistä uimareista uimarin ja painaa ilmoittamisnappia. Tämän jälkeen järjestelmä lisää kyseisen sarjan tietokantaan uimarin lisenssinumeron sekä tulostaa näytölle taulukkoon ilmoitetun uimarin tiedot.

Poikkeukset: Jos uimari on jo aikaisemmin ilmoitettu samaan sarjaan, järjestelmä ei anna ilmoittaa uimaria ja antaa siitä virheilmoituksen.

7 Ilmoittautumisen peruminen

Yhteenveto: Käyttäjä poistaa aikaisemmin ilmoitetun uimarin tiedot käsiteltävästä sarjasta.

Kuvaus: Käyttäjä valitsee alasvetovalikosta peruutettavan ilmoittautumisen. Tämän jälkeen käyttäjä painaa peruutusnappia ja järjestelmä poistaa kyseisen uimarin tiedot sarjasta ja järjestää muut ilmoitetut uimarit uudelleen ilmoittautumisjärjestyksessä.

Poikkeukset: Jos sarjaan ei ole ilmoitettu uimareita, ei peruutusalasvetovalikossa ole valittavissa mitään.

8 Tuomareiden ilmoittaminen

Yhteenveto: Käyttäjä määrittää kuinka monta tuomaria halutaan ilmoittaa ja mihin sarjaan tuomarit ilmoitetaan.

Kuvaus: Käyttäjä valitsee alasvetovalikosta haluamansa sarjan ja tuomarien määrän kolmesta seitsemään. Tämän jälkeen käyttäjä painaa siirtymisnappia ja järjestelmä siirtyy kyseisen sarjan ilmoittautumissivulle.

Poikkeukset: 1. Jos sarjoja ei ole aikaisemmin jo perustettu, ei sarjoja ole valittavissa alasvetovalikossa.

2. Jos tuomarien määrää ei määritetä, järjestelmä valitsee automaattisesti ilmoitettavien tuomareiden määräksi seitsemän.

9 Tuomarin ilmoittaminen sarjaan.

Yhteenveto: Käyttäjä ilmoittaa valitsemansa määrän tuomareita aikaisemmin valittuun sarjaan.

Kuvaus: Käyttäjä ilmoittaa jokaisen tuomarin sukunimen, etunimen ja seuran. Järjestelmä tallentaa tiedot painamalla tallennusnappia ja tämän jälkeen järjestelmä palaa tuomareiden ilmoittamissivulle.

Poikkeukset: Jos sarjaan on jo ilmoitettu tuomarit, järjestelmä tulostaa näytölle aikaisemmin ilmoitetuista tuomareista listan jonka voi hyväksyä tai järjestelmää voi pyytää tyhjentämään aikaisemmin ilmoitettujen tuomareiden tiedot.

10 Kilpailun aloittaminen

Yhteenveto: Käyttäjä valitsee kilpailtavan sarjan ja ilmoittaa tuomareiden antamat pisteet järjestelmään.

Kuvaus: Käyttäjä valitsee alasvetovalikosta kilpailtavan sarjan ja siirtyy ilmoittamaan pisteitä painamalla aloittamisnappia.

Jokaisen uitavan kuvion ensimmäisen uimarin kohdalla järjestelmä kysyy aloittavaa uimaria. Tämä tapahtuu määritellään valitsemalla uimari alasvetovalikosta. Jokaisen uimarin kohdalla ilmoitetaan kunkin tuomarin antamat pisteet ja virhepisteet tekstikenttiin. Järjestelmä siirtyy seuraavaan uimariin painamalla siirtymisnappia.

Järjestelmä käy läpi kaikki uimarit kussakin neljässä kuviossa, jonka jälkeen järjestelmä ilmoittaa kilpailun loppuneen. Painamalla hyväksymisnappia järjestelmä laskee lopulliset pisteet ja valmistelee tulokset. Tulosten luomisen jälkeen järjestelmä siirtyy takaisin kilpailun aloittamissivulle.

Poikkeukset: Jos yhtään sarjaa ei ole perustettu, ei järjestelmä tulosta alasvetovalikkoon yhtään valittavaa sarjaa.

Jos valittuun sarjaan ei ole ilmoitettu uimareita, kilpailua ei voida aloittaa.

Jos järjestelmään ei ole ilmoitettu tuomareita, järjestelmä ilmoittaa tuomareiden puuttumisesta ja ehdottaa tuomareiden ilmoittamista.

11 Tulosteiden tarkastelu

Yhteenveto: Käyttäjä tarkastelee osallistujaluetteloita sekä lopputuloksia.

Kuvaus: Käyttäjä valitsee halutun sarjan osallistujaluettelon/lähtölistan tai lopputuloksen painamalla sille varattua nappia. Tämän jälkeen järjestelmä avaa uuteen ikkunaan valitun tulosteen.

Poikkeukset: Jos perustettuun sarjaan ei ole ilmoitettu uimareita, ei osallistujaluetteloa/lähtölistaa ei sarjalle ole näkyvissä omaa painiketta.

Jos sarjaa ei ole vielä kilpailtu tai järjestelmä ei ole vielä laskenut kyseisen sarjan pisteitä, ei järjestelmä tarjoa kyseisen sarjan lopputuloksille painiketta.

12 Järjestelmän sulkeminen

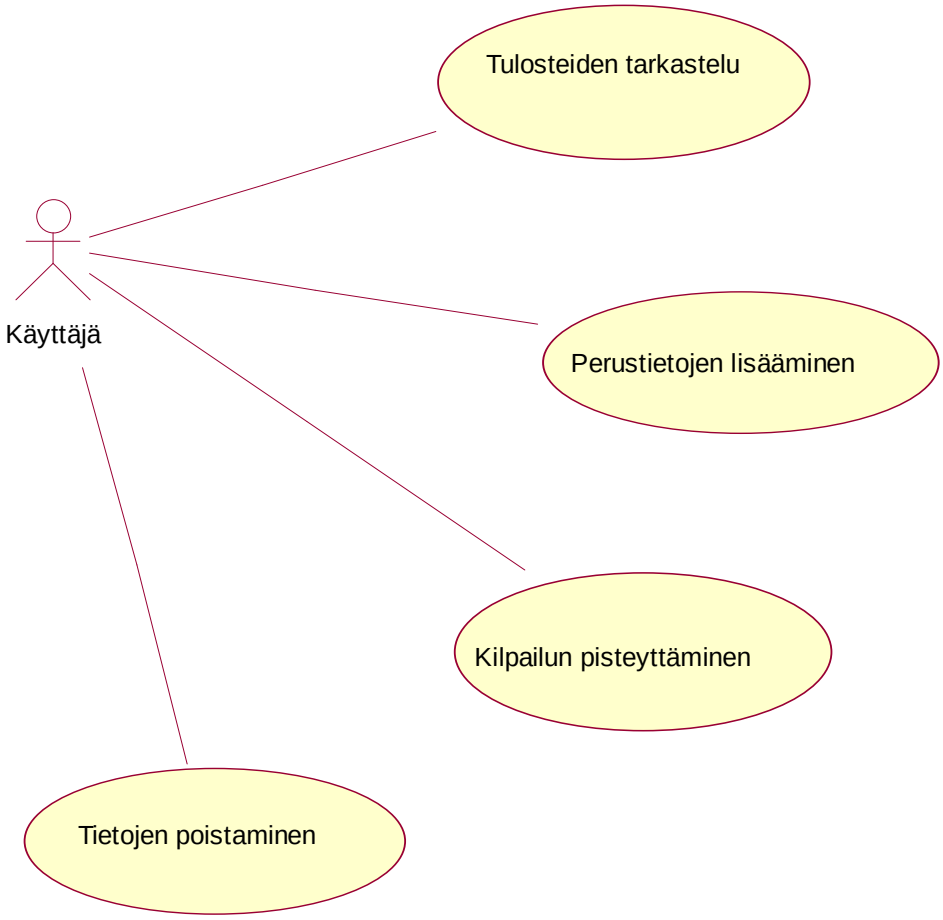
Yhteenveto: Käyttäjä sulkee järjestelmän.

Kuvaus: Käyttäjän painaessa sulkemisnappia, järjestelmä vaatii käyttäjän toimille vahvistuksen. Hyväksymällä sulkemisen, järjestelmä sulkee avoimen tietokantayhteyden ja palvelinyhteyden. Painamalla ei järjestelmä siirtyy etusivulle.

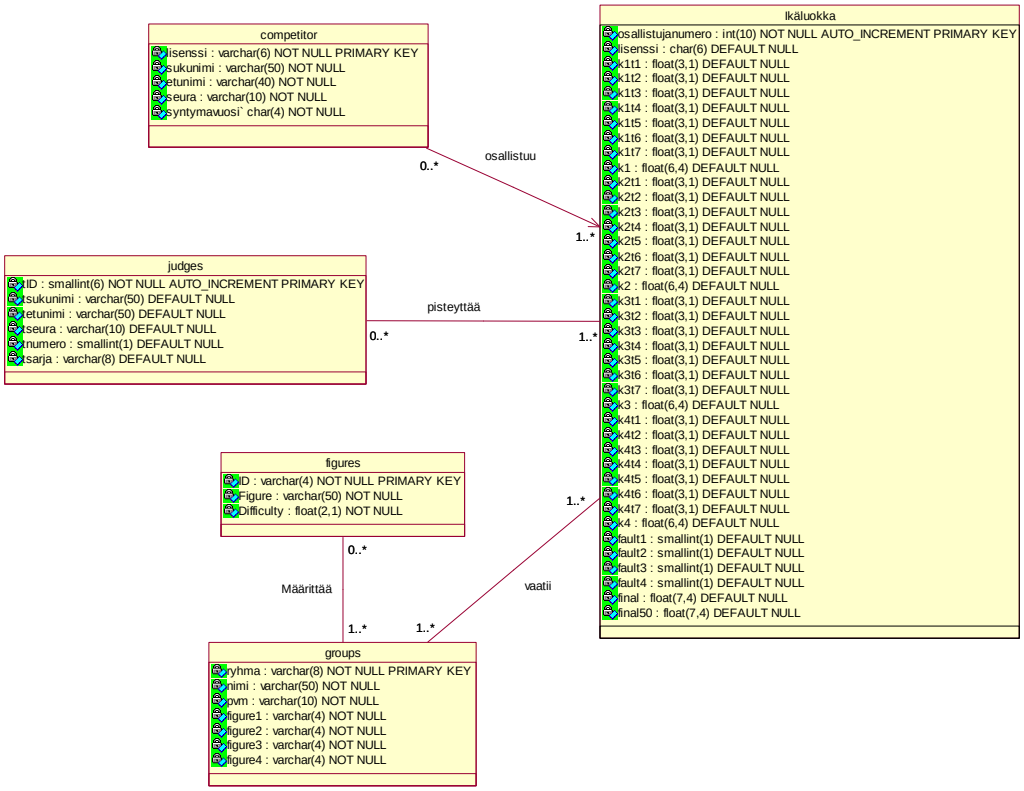
Poikkeukset: Ei

Liite 4: Kaaviot:

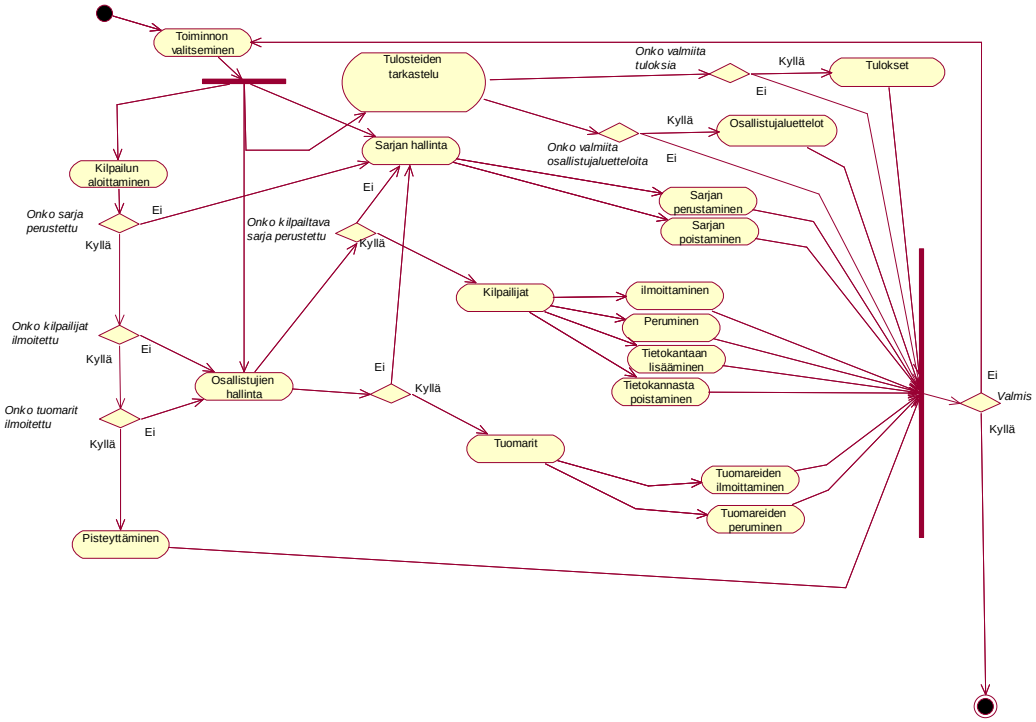
Käyttötapauskaavio



Luokkakaavio



Kulkukaavio:



Liite 5: Esimerkki ohjelmistokoodista

Alla olevassa ohjelmiston koodin osassa käydään läpi uimarin poistamista kilpailusta. Eli ensin järjestelmä ottaa yhteyden tietokantaan, minkä jälkeen se tarkistaa onko sarjassa kilpailija jota yritetään poistaa. Jos sarjasta ei löydy poistettavaa kilpailijaa, ei järjestelmä tee käytännössä mitään. Jos poistettava uimari löytyy sarjasta joko esiuimarina tai kilpailijana, järjestelmä suorittaa poiston ja lataa näkyville uuden korjatun luettelon ilmoitetuista uimareista.

```
<?php
$type="mysql"; $user="root"; $pwd=""; $host="localhost"; $dbname="taitouinti";
$dbh = new PDO('mysql:host=localhost;dbname=taitouinti',"root","");
$poisto=$_POST["lisenssi"];
$sarja=$_POST["sarja"];
$tietokantapoisto=$_POST["poisto"];

$link = mysql_connect("localhost", "root", "");
if (!$link){die('ei yhteytta:' .mysql_error());}
mysql_select_db("taitouinti");

if ($sarja == ag12) {
if ($tietokantapoisto !== NULL){
$sql = $dbh->prepare('DELETE from competitor WHERE lisenssi = :lisenssi');
$sql->bindParam(":lisenssi",$tietokantapoisto);
$ok = $sql->execute();
if(!$ok) {print_r($sql->errorInfo());}}

$tarkistus= mysql_query("SELECT osallistujanumero from ag12 where lisenssi = '$poisto'");
$tarkistus2= mysql_fetch_row($tarkistus);
if ($tarkistus2[0] == NULL){
$sql = $dbh->prepare('DELETE from preswimmer WHERE lisenssi = :lisenssi and sarja = "ag12"');
$sql->bindParam(":lisenssi",$poisto);
$ok = $sql->execute();
if(!$ok) {print_r($sql->errorInfo());}
```

```
$kysely=mysql_query("SELECT lisenssi from preswimmer where sarja = 'ag12'", $link);
$rivit=mysql_num_rows($kysely);
$i=1;
for(mysql_query("delete from preswimmer where sarja = 'ag12'", $link); $i<=$rivit; $i++){
mysql_query("ALTER TABLE preswimmer auto_increment = 1");
$osallistujat=mysql_fetch_row($kysely);
mysql_query("INSERT INTO preswimmer (lisenssi, sarja) values ('$osallistujat[0]', 'ag12')");}

header("Location: http://localhost/ag12.php");}

else{
$sql = $dbh->prepare('DELETE from ag12 WHERE lisenssi = :lisenssi');
$sql->bindParam(":lisenssi",$poisto);
$ok = $sql->execute();
if(!$ok) {print_r($sql->errorInfo());}

$kysely=mysql_query("SELECT lisenssi from ag12", $link);
$rivit=mysql_num_rows($kysely);
$i=1;
for(mysql_query("delete from ag12", $link); $i<=$rivit; $i++){
mysql_query("ALTER TABLE ag12 auto_increment = 1");
$osallistujat=mysql_fetch_row($kysely);
mysql_query("INSERT INTO ag12 (lisenssi) values ('$osallistujat[0]')");}
if ($link) {header("Location: http://localhost/ag12.php");}}}

(Koodi jatkuu muiden sarjojen osalta samanlaisena.)

?>
```